

Informix I-Spy User Manual

Version 1.0
January 1999
Part No. 000-5317

Published by INFORMIX® Press

Informix Corporation
4100 Bohannon Drive
Menlo Park, CA 94025-1032

© 1999 Informix Corporation. All rights reserved. The following are trademarks of Informix Corporation or its affiliates:

Answers OnLine™; CBT Store™; C-ISAM®; Client SDK™; ContentBase™; Cyber Planet™; DataBlade®; Data Director™; Decision Frontier™; Dynamic Scalable Architecture™; Dynamic Server™; Dynamic Server™, Developer Edition™; Dynamic Server™ with Advanced Decision Support Option™; Dynamic Server™ with Extended Parallel Option™; Dynamic Server™ with MetaCube® ROLAP Option; Dynamic Server™ with Universal Data Option™; Dynamic Server™ with Web Integration Option™; Dynamic Server™, Workgroup Edition™; FastStart™; 4GL for ToolBus™; If you can imagine it, you can manage itSM; Illustra®; INFORMIX®; Informix Data Warehouse Solutions... Turning Data Into Business Advantage™; INFORMIX®-Enterprise Gateway with DRDA®, Informix Enterprise Merchant™; INFORMIX®-4GL; Informix-JWorks™; InformixLink®, Informix Session Proxy™; InfoShelf™; Interforum™; I-Spy™; Mediazation™; MetaCube®; NewEra™; ON-Bar™; OnLine Dynamic Server™; OnLine for NetWare®, OnLine / Secure Dynamic Server™; OpenCase®, ORCA™; Regency Support®; Solution Design LabsSM; Solution Design ProgramSM; SuperView®; Universal Database Components™; Universal Web Connect™; ViewPoint®, Visionary™; Web Integration Suite™. The Informix logo is registered with the United States Patent and Trademark Office. The DataBlade logo is registered with the United States Patent and Trademark Office.

Documentation Team: Mark Jeske, Mary Kraemer, Scott Nash, Barbara Nomiyama, Patrice O'Neill, Matt Veal, Keldyn West, Eileen Wollam

GOVERNMENT LICENSE RIGHTS

Software and documentation acquired by or for the US Government are provided with rights as follows:
(1) if for civilian agency use, with rights as restricted by vendor's standard license, as prescribed in FAR 12.212;
(2) if for Dept. of Defense use, with rights as restricted by vendor's standard license, unless superseded by a negotiated vendor license, as prescribed in DFARS 227.7202. Any whole or partial reproduction of software or documentation marked with this legend must reproduce this legend.

Table of Contents

Introduction

In This Introduction	3
About This Manual	3
Types of Users	4
Software Dependencies	4
Assumptions About Your Locale.	5
Documentation Conventions	5
Typographical Conventions	6
Icon Conventions	7
Command-Line Conventions	7
Sample-Code Conventions.	10
Additional Documentation	10
On-Line Manuals	11
Printed Manuals	11
Error Message Documentation	11
Documentation Notes, Release Notes, Machine Notes	12
Compliance with Industry Standards	12
Informix Welcomes Your Comments	13

Chapter 1

Informix I-Spy

In This Chapter	1-3
What Is I-Spy?	1-3
I-Spy Features	1-4
Current Limitations	1-5

Chapter 2

I-Spy Configuration

In This Chapter	2-3
I-Spy Directory Structure	2-3
activity	2-4
bin	2-4
etc	2-4
gls	2-5
lib	2-5
log	2-5
msg	2-5
release	2-5
sql	2-6
web	2-6
Environment Variable Settings	2-6
ISPY_DIR	2-7
ISPY_CONFIGFILE	2-7
INFORMIXDIR and INFORMIXSQLHOSTS	2-8
Shared-Library Search Path	2-8
PATH	2-9
Prepare the ISCONFIG Configuration File	2-10
Prepare the sqlhosts File	2-11
Start the ISPY Daemon	2-12
Configuring Clients for I-Spy	2-13
Configuring INFORMIX-Connect	2-13
What to Do Next?	2-14

Chapter 3

Configuration File

In This Chapter	3-3
General Configuration-File Syntax	3-3
Rules Overview	3-4
Rule Execution Order	3-6
Regular-Expression Syntax	3-6
ACTION	3-8
ACTION ENDSRSS	3-9
ACTION ERROR.	3-9
ACTION EXECSQL	3-10
ACTION EXECUTE.	3-10
ACTION EXPLAINTAGS.	3-12
ACTION NOLOG	3-13

ACTIVITY_CAPTURE	3-14
ACTIVITY_DBNAME	3-15
ACTIVITY_DIR	3-15
ACTIVITY_ID	3-16
ACTIVITY_INFORMIXSERVER	3-16
DEBUGFILE	3-17
DEBUGLEVEL	3-17
ECOST	3-18
EROWS	3-18
EXPLAINTAGS	3-19
HOST	3-20
ISPYSERVER	3-21
ITIME	3-22
LOGFILE	3-22
PHASE	3-23
QTIME	3-23
ROWS	3-25
RULE	3-25
SHADOWSERVER	3-27
SQL	3-27
SQLLENGTH	3-28
SQLTRIM	3-28
USER	3-30
Advanced Configuration Options	3-30
Monitoring Multiple Database Servers	3-31
Configuring I-Spy with No Changes in the Client Environment	3-32
Multiple I-Spy Configurations, One Activity Database?	3-33

Chapter 4

Configuring the Web Interface

In This Chapter	4-3
Preparing the web.cfg Web Configuration File	4-4
Configuring Web Server Directory Access	4-5
Adding a URL Alias	4-5
Allowing CGI Execution and Directory Access	4-6
Granting Access to the Activity Database	4-7
Testing the Web Interface	4-7
Configuration of Basic Web Security	4-8

Copy the Informix I-Spy User Manual	4-9
Custom Reports Link on the Reports Page	4-10
Configuration for Netscape FastTrack Server	4-12
Create New Document Directory	4-12
Activate the CGI File Type	4-13

Chapter 5 I-Spy Utilities

In This Chapter	5-3
isexplain: Load Explain Text	5-3
isload: Load Files into the Activity Database	5-5
ispy: I-Spy Daemon Process	5-6

Chapter 6 Activity Database Layout

In This Chapter	6-3
Creating the Activity Database	6-3
Determine Database Location	6-4
Determine act_sqlstmt Column Type	6-4
Prepare the ispy.sql File	6-5
Add Storage-Specific Syntax	6-5
Run ispy.sql	6-6
Grant Connect Privilege	6-6
ISPY_SESSION	6-8
ISPY_ACTIVITY	6-9
ISPY_EXPLAIN	6-13
ISPY_SQLTYPE	6-15

Chapter 7 Using the Web Interface

In This Chapter	7-3
The Main Window	7-4
The Startup Menu Option	7-5
The Shutdown Menu Option	7-6
The Config File Menu Option	7-7
The Clients Menu Option	7-8
The Reports Menu Option	7-10
The View Log Menu Option	7-12
The User Manual Menu Option	7-12

Chapter 8	Common Questions	
	In This Chapter	8-3
	Common Questions About I-Spy	8-3
	How do I prevent users from returning more than 1000 rows in a single query?	8-3
	How do I send an email to the database administrator when a query has returned more than 1000 rows without ending the query?.	8-4
	How do I prevent all users except informix from returning more than 1000 rows?	8-6
	How do I assign a default PDQ priority for all users?.	8-6
	How do I check the syntax of the rules in my configuration file without shutting down and starting up I-Spy?	8-7

Index

Introduction

About This Manual	3
Types of Users	4
Software Dependencies	4
Assumptions About Your Locale	5
Documentation Conventions	5
Typographical Conventions	6
Icon Conventions	7
Command-Line Conventions	7
How to Read a Command-Line Diagram	9
Sample-Code Conventions	10
Additional Documentation	10
On-Line Manuals	11
Printed Manuals	11
Error Message Documentation	11
Documentation Notes, Release Notes, Machine Notes	12
Compliance with Industry Standards	12
Informix Welcomes Your Comments	13

In This Introduction

This introduction provides an overview of the information in this manual and describes the conventions it uses.

About This Manual

Informix I-Spy is a tool that provides new and more effective ways to quantify, manage, and control utilization of data warehouses for database administrators (DBAs) and data warehouse architects.

Informix I-Spy lets DBAs view the SQL being executed and the data being accessed. It reports execution time and the quantity of data being returned. You can use the product to identify inefficient queries and pinpoint users who might need help or additional training.

Informix I-Spy has standard utilization reports that can help you avoid capacity problems by providing the precise data needed for capacity planning. Warehouse architects can use the information collected by I-Spy to help refine their warehouse model for its next version. They can analyze query plans to develop optimal indexing strategies, and determine which fact and dimension tables are really being used.

This manual provides detailed instructions to configure and use Informix I-Spy to monitor and control SQL statements submitted to an Informix database server.

Types of Users

This manual is written for the following users:

- Database administrators
- Database developers
- Data warehouse architects

This manual assumes that you have the following background:

- A working knowledge of your computer, your operating system, and the utilities that your operating system provides
- Some experience working with relational databases or exposure to database concepts
- Some experience with database server administration, operating-system administration, or network administration

If you have limited experience with relational databases, SQL, or your operating system, refer to the *Getting Started* manual for your database server for a list of supplementary titles.

Software Dependencies

This manual assumes that you are using one of the following database servers:

- Informix Dynamic Server with Advanced Decision Support and Extended Parallel Options, Version 8.11
- Informix Dynamic Server with Advanced Decision Support and Extended Parallel Options, Version 8.2
- Informix Dynamic Server, Version 7.2
- Informix Dynamic Server, Version 7.3
- Informix Dynamic Server, Workgroup Edition, Version 7.3

Assumptions About Your Locale

Informix products can support many languages, cultures, and code sets. All culture-specific information is brought together in a single environment, called a Global Language Support (GLS) locale.

The examples in this manual are written with the assumption that you are using the default locale, **en_us.8859-1**. This locale supports U.S. English format conventions for dates, times, and currency. In addition, this locale supports the ISO 8859-1 code set, which includes the ASCII code set plus many 8-bit characters such as é, è, and ñ.

If you plan to use nondefault characters in your data or your SQL identifiers, or if you want to conform to the nondefault collation rules of character data, you need to specify the appropriate nondefault locale.

For instructions on how to specify a nondefault locale, additional syntax, and other considerations related to GLS locales, see the *Informix Guide to GLS Functionality*.

Documentation Conventions

This section describes the conventions that this manual uses. These conventions make it easier to gather information from this and other volumes in the documentation set.

The following conventions are discussed:

- Typographical conventions
- Icon conventions
- Command-line conventions
- Sample-code conventions

Typographical Conventions

This manual uses the following conventions to introduce new terms, illustrate screen displays, describe command syntax, and so forth.

Convention	Meaning
KEYWORD	All primary elements in a programming language statement (keywords) appear in uppercase letters in a serif font.
<i>italics</i> italics <i>italics</i>	Within text, new terms and emphasized words appear in italics. Within syntax and code examples, variable values that you are to specify appear in italics.
boldface boldface	Names of program entities (such as classes, events, and tables), environment variables, file and pathnames, and interface elements (such as icons, menu items, and buttons) appear in boldface.
<code>monospace</code> <code>monospace</code>	Information that the product displays and information that you enter appear in a monospace typeface.
KEYSTROKE	Keys that you are to press appear in uppercase letters in a sans serif font.
◆	This symbol indicates the end of one or more product- or platform-specific paragraphs.
→	This symbol indicates a menu item. For example, “Choose Tools→Options ” means choose the Options item from the Tools menu.



Tip: When you are instructed to “enter” characters or to “execute” a command, immediately press RETURN after the entry. When you are instructed to “type” the text or to “press” other keys, no RETURN is required.

Icon Conventions

Comment icons identify three types of information, as the following table describes. This information always appears in *italics*.

Icon	Label	Description
	<i>Warning:</i>	Identifies paragraphs that contain vital instructions, cautions, or critical information
	<i>Important:</i>	Identifies paragraphs that contain significant information about the feature or operation that is being described
	<i>Tip:</i>	Identifies paragraphs that offer additional details or shortcuts for the functionality that is being described

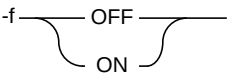
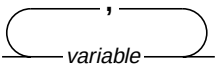
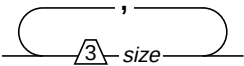
Command-Line Conventions

This section defines and illustrates the format of commands that are available in Informix products. These commands have their own conventions, which might include alternative forms of a command, required and optional parts of the command, and so forth.

Each diagram displays the sequences of required and optional elements that are valid in a command. A diagram begins at the upper-left corner with a command. It ends at the upper-right corner with a vertical line. Between these points, you can trace any path that does not stop or back up. Each path describes a valid form of the command. You must supply a value for words that are in *italics*.

You might encounter one or more of the following elements on a command-line path.

Element	Description
command	This required element is usually the product name or other short word that invokes the product or calls the compiler or preprocessor script for a compiled Informix product. It might appear alone or precede one or more options. You must spell a command exactly as shown and use lowercase letters.
<i>variable</i>	A word in italics represents a value that you must supply, such as a database, file, or program name. A table following the diagram explains the value.
-flag	A flag is usually an abbreviation for a function, menu, or option name, or for a compiler or preprocessor argument. You must enter a flag exactly as shown, including the preceding hyphen.
.ext	A filename extension, such as .sql or .cob , might follow a variable that represents a filename. Type this extension exactly as shown, immediately after the name of the file. The extension might be optional in certain products.
(. , ; + * - /)	Punctuation and mathematical notations are literal symbols that you must enter exactly as shown.
' '	Single quotes are literal symbols that you must enter as shown.
Privileges p. 5-17 Privileges	A reference in a box represents a subdiagram. Imagine that the subdiagram is spliced into the main diagram at this point. When a page number is not specified, the subdiagram appears on the same page.
— ALL —	A shaded option is the default action.
→ →	Syntax within a pair of arrows indicates a subdiagram.
—	The vertical line terminates the command.

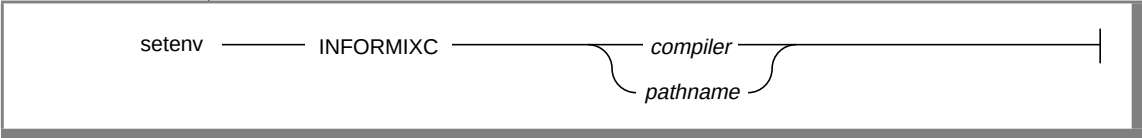
Element	Description
	A branch below the main path indicates an optional path. (Any term on the main path is required, unless a branch can circumvent it.)
	A loop indicates a path that you can repeat. Punctuation along the top of the loop indicates the separator symbol for list items.
	A gate ($\triangle 3$) on a path indicates that you can only use that path the indicated number of times, even if it is part of a larger loop. You can specify <i>size</i> no more than three times within this statement segment.

(2 of 2)

How to Read a Command-Line Diagram

Figure 1 shows a command-line diagram that uses some of the elements that are listed in the previous table.

Figure 1
Example of a Command-Line Diagram



To construct a command correctly, start at the top left with the command. Follow the diagram to the right, including the elements that you want. The elements in the diagram are case sensitive.

Figure 1 illustrates the following steps:

1. Type `setenv`.
2. Type `INFORMIX`.
3. Supply either a compiler name or a pathname.
After you choose *compiler* or *pathname*, you come to the terminator. Your command is complete.
4. Press RETURN to execute the command.

Sample-Code Conventions

Examples of SQL code occur throughout this manual. Except where noted, the code is not specific to any single Informix application development tool. If only SQL statements are listed in the example, they are not delimited by semicolons. For instance, you might see the code in the following example:

```
CONNECT TO stores_demo
...

DELETE FROM customer
      WHERE customer_num = 121
...

COMMIT WORK
DISCONNECT CURRENT
```

To use this SQL code for a specific product, you must apply the syntax rules for that product. For example, if you are using DB-Access, you must delimit multiple statements with semicolons. If you are using an SQL API, you must use EXEC SQL at the start of each statement and a semicolon (or other appropriate delimiter) at the end of the statement.



Tip: *Ellipsis points in a code example indicate that more code would be added in a full application, but it is not necessary to show it to describe the concept being discussed.*

For detailed directions on using SQL statements for a particular application development tool or SQL API, see the manual for your product.

Additional Documentation

For additional information, you might want to refer to the following types of documentation:

- On-line manuals
- Printed manuals
- Error message documentation
- Documentation notes, release notes, and machine notes
- Related reading

On-Line Manuals

An Answers OnLine CD that contains Informix manuals in electronic format is provided with your Informix products. You can install the documentation or access it directly from the CD. For information about how to install, read, and print on-line manuals, see the installation insert that accompanies Answers OnLine.

Printed Manuals

To order printed manuals, call 1-800-331-1763 or send email to moreinfo@informix.com. Please provide the following information when you place your order:

- The documentation that you need
- The quantity that you need
- Your name, address, and telephone number

Error Message Documentation

Informix software products provide ASCII files that contain all of the Informix error messages and their corrective actions.

Tip: *The error message utilities described in this section are not shipped as part of I-Spy. They are part of your Informix database server product.*



UNIX

To read error messages and corrective actions on UNIX, use one of the following utilities.

Utility	Description
finderr	Displays error messages on line
rofferr	Formats error messages for printing



WIN NT

To read error messages and corrective actions in Windows environments, use the **Informix Find Error** utility. To display this utility, choose **Start→Programs→Informix** from the Task Bar. ♦

UNIX

Instructions for using the preceding utilities are available in Answers OnLine. Answers OnLine also provides a listing of error messages and corrective actions in HTML format.

Documentation Notes, Release Notes, Machine Notes

In addition to printed documentation, the following sections describe the on-line files that supplement the information in this manual. Please examine these files before you begin using your database server. They contain vital information about application and performance issues.

On UNIX platforms, the following on-line files may appear in the **\$INFORMIXDIR/release** directory. Replace x.y in the filenames with the version number of I-Spy.

On-Line File	Purpose
ISPYDOCx.y	The documentation-notes file for your version of this manual describes topics that are not covered in the manual or that were modified since publication.
ISPYx.y	The release-notes file describes feature differences from earlier versions of Informix products and how these differences might affect current products. This file also contains information about any known problems and their workarounds.



Compliance with Industry Standards

The American National Standards Institute (ANSI) has established a set of industry standards for SQL. Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL CAE (common applications environment) standards.

Informix Welcomes Your Comments

Let us know what you like or dislike about our manuals. To help us with future versions of our manuals, we want to know about any corrections or clarifications that you would find useful. Include the following information:

- The name and version of the manual that you are using
- Any comments that you have about the manual
- Your name, address, and phone number

Write to us at the following address:

Informix Software, Inc.
SCT Technical Publications Department
4100 Bohannon Drive
Menlo Park, CA 94025

If you prefer to send electronic mail, our address is:

`doc@informix.com`

The **doc** alias is reserved exclusively for reporting errors and omissions in our documentation.

We appreciate your suggestions.

Informix I-Spy

What Is I-Spy?	1-3
I-Spy Features	1-4
Current Limitations	1-5

In This Chapter

Informix I-Spy provides a query-management facility to monitor and control SQL statements submitted to an Informix database server. I-Spy transparently intercepts all SQL statements submitted to a database server and saves them in an activity database. Database administrators and developers can use this historical collection of SQL activity to tune the database server and gain a precise understanding of how clients use the system. I-Spy also provides a rule system by which you can place constraints such as maximum query time or maximum rows returned on individual SQL statements.

This manual provides detailed instructions for configuring and using I-Spy to monitor the SQL activity of your database servers.

What Is I-Spy?

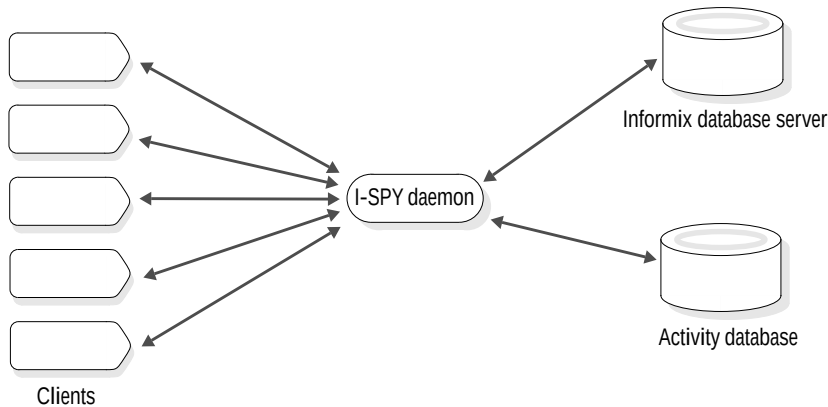
I-Spy is a daemon process that looks like a database server to client applications. It listens on a TCP port for connections and routes those connections and all associated traffic to an actual Informix database server. For the remainder of this manual, this Informix database server is referred to as the *shadow server*. While it is routing the network traffic, the shadow server decodes all messages and logs the SQL statements and associated statistics to an activity database. An I-Spy configuration consists of the following components:

- The Informix database client
- The I-Spy daemon process that listens for client connections
- The Informix database server that the client wants to connect to
- The Informix database server that contains the I-Spy activity database used to record all SQL statements submitted by clients

Each of these components can be located on a different computer, or they can be combined on the same computer.

[Figure 1-1](#) illustrates these components.

Figure 1-1
An I-Spy Configuration



Each of these components can be run on the same computer, or on separate computers. [Chapter 2, “I-Spy Configuration”](#) discusses the various configuration options.

I-Spy Features

I-Spy is a query-management tool that provides all of the following features:

- I-Spy installs transparently into an existing environment.
No changes need to be made to either the client applications or the database server for I-Spy to operate. The only change necessary for the client is a different value for **INFORMIXSERVER**. Depending on how I-Spy is configured, even the change to the client environment is unnecessary. Configuration is covered in [Chapter 2](#).

- I-Spy logs SQL statement text and various statistics related to the statement such as total time of execution, number of rows returned, length of the returned row, and so on.
- I-Spy logs the start and end time of all client connections.
An Informix activity database is used to record all logged information. This allows you to access the logged information using SQL.
- I-Spy can process explain files generated by the database server to add the explain text into the activity database.
This allows you to see exactly how the database server processed a specific query.
- A rules system allows constraints to be placed on individual SQL statements.
The rules allow actions to be performed based on total query time, number of rows returned, or the actual text of the SQL statement.
- A Web interface administers I-Spy.
This interface allows you to perform all aspects of I-Spy administration as well as run a number of predefined reports against the activity database.

Current Limitations

I-Spy does not currently support the following configurations:

- Clients created prior to release 6.0 of the Informix development tools
- Clients that use shared memory or stream pipes for communications to the database server
Only network connections are supported.
- Connection multiplexing enabled in the **sqlhosts** file
- Database servers prior to 7.1x
- Database servers with the Universal Data Option
- Clients that communicate with the database server using the XA protocol
- Database servers performing distributed SQL

I-Spy Configuration

I-Spy Directory Structure	2-3
activity.	2-4
bin	2-4
etc	2-4
glb	2-5
lib	2-5
log	2-5
msg.	2-5
release	2-5
sql	2-6
web.	2-6
Environment Variable Settings	2-6
ISPY_DIR.	2-7
ISPY_CONFIGFILE	2-7
INFORMIXDIR and INFORMIXSQLHOSTS.	2-8
Shared-Library Search Path	2-8
PATH	2-9
Prepare the ISCONFIG Configuration File	2-10
Prepare the sqlhosts File	2-11
Start the ISPY Daemon	2-12
Configuring Clients for I-Spy	2-13
Configuring INFORMIX-Connect	2-13
What to Do Next?	2-14

In This Chapter

Before you can begin using I-Spy, you must configure it for your environment and start to run the I-Spy daemon process. Once the daemon process is running, clients will connect to this process by changing their **INFORMIXSERVER** environment variable setting.

This chapter describes the following tasks:

- Familiarization with the I-Spy directory structure
- Setting the necessary environment variables for I-Spy
- Preparing the **ISCONFIG** configuration file
- Preparing the **sqlhosts** file
- Starting the **ispy** daemon
- Configuring clients to use I-Spy
- What To Do Next?

I-Spy Directory Structure

After you follow the installation instructions that accompanied your I-Spy product, you should have a new directory tree containing all of the files required for I-Spy to operate. You should have an owner and group named **informix** on your computer, and all of the files should be owned by **informix** and have a group of **informix**. If this is not the case, review the installation instructions to confirm that you followed them correctly.



Warning: If the computer where you installed I-Spy already has existing Informix products installed on it, make sure that you have installed I-Spy into a separate directory. If you install I-Spy into the same directory as existing Informix products, I-Spy might fail to operate, and you might introduce problems with your existing products. For example, if you have an Informix database server installed in **/usr/informix**, you might want to install I-Spy in **/usr/ispyspy**, or even **/usr/informix/ispyspy**, but not directly in the **/usr/informix** directory.

The I-Spy installation script creates the directory **./ispyspytmp**. Various files for I-Spy are located in this directory. Never edit, rename, or remove any files in this directory, and never remove or rename the directory itself. If you do I-Spy will fail to operate.

The I-Spy product installation includes the following directories.

activity

The **activity** directory is an empty directory intended to hold activity files if I-Spy is configured to capture activity to the file system. The **ACTIVITY_CAPTURE** parameter in the configuration file is used to specify where I-Spy places activity information.

bin

The **bin** directory contains all the I-Spy utilities: **ispyspy**, **isload**, and **isexplain**. The **ispyspy** executable is the I-Spy daemon process that intercepts all client activity and saves it to the activity database. The **isload** executable is used to load files from the file system into the activity database. The **isexplain** executable is used to process **sqexplain.out** files created by Informix database servers and to insert the explain text for SQL statements into the activity database.

etc

The **etc** directory contains a sample I-Spy configuration file named **isconfig.std** and a sample sqlhosts file named **sqlhosts.std**. The directory also contains some other files used only during product installation. Unless told otherwise, the **ispyspy** executable looks for a configuration file named **isconfig** in this directory. A later section in this chapter describes how to specify a different location and filename for the configuration file.

gls

The **gls** directory contains the GLS files necessary for the I-Spy utilities. The use of these files is entirely transparent to the operation of I-Spy, and you can safely ignore this directory.

lib

The **lib** directory contains the shared library files necessary for the I-Spy utilities to operate. It is necessary to configure your environment so that the operating system looks into this directory for shared libraries before it runs any of the I-Spy utilities. This topic is discussed later in this chapter.

log

The **log** directory is an empty directory you can use to hold the I-Spy log file and the I-Spy debug file if debugging is enabled.

msg

The **msg** directory contains the Informix message files required for the I-Spy utilities. The use of these files is entirely transparent to the operation of I-Spy, and you can safely ignore this directory.

release



Important: The **release** directory contains any last-minute information on I-Spy that could not be added to this documentation. It is very important that you read the files in this directory before you continue with the I-Spy configuration.

sql

The **sql** directory contains **ispy.sql**, the schema file used to create the activity database. A number of **rpt_*.sql** files are also contained in this directory. Each of these files contains a single SQL statement used to create a report from the activity database. Each file corresponds to a report available in the Web interface for I-Spy. You cannot use these files directly because they contain placeholders for various SQL WHERE clause values, such as user names and dates. You must substitute appropriate values before you run them against the activity database.

web

The **web** directory contains all of the files required for the Web interface to I-Spy. Chapter 4 describes how to configure and operate the Web interface. It is recommended that you successfully configure I-Spy and make sure it is operating correctly before you attempt to use the Web interface.

Environment Variable Settings

A number of environment variables are required to start the I-Spy daemon process or execute any of the I-Spy utilities. The following environment variables are covered in this section:

- **ISPY_DIR**
- **ISPY_CONFIGFILE**
- **INFORMIXDIR** and **INFORMIXSQLHOSTS**
- Shared-library search path
- **PATH**



Important: These environment variable settings are only required on the computer where I-Spy has been installed to execute and administer I-Spy. They are not required by client applications that connect through I-Spy to an Informix database server.

ISPY_DIR

ISPY_DIR is a required environment variable and contains the full path to the directory where you have installed I-Spy. For example, if you installed I-Spy in **/usr/isy**, you would set **ISPY_DIR** as follows:

```
$ ISPY_DIR=/usr/isy
$ export ISPY_DIR
```

ISPY_CONFIGFILE

ISPY_CONFIGFILE is an optional environment variable that contains the full path to the configuration file you want I-Spy to use. I-Spy looks for the configuration file it will use in the following order:

1. If the command-line option **-c filename** is present, I-Spy uses *filename* as the configuration file.
2. If the **ISPY_CONFIGFILE** environment variable is set, I-Spy uses it for the configuration file.
3. If neither the **-c filename** command line-option is present, nor the **ISPY_CONFIGFILE** environment variable is set, I-Spy looks for the file **\$ISPY_DIR/etc/isconfig**.

If your configuration file is named **/usr/isy/etc/isconfig.server1**, you would set **ISPY_CONFIGFILE** as follows:

```
$ ISPY_CONFIGFILE=/usr/isy/etc/isconfig.server1
$ export ISPY_CONFIGFILE
```

INFORMIXDIR and INFORMIXSQLHOSTS

The **INFORMIXDIR** and **INFORMIXSQLHOSTS** environment variables are only used by I-Spy to locate an **sqlhosts** file. Both of these variables are optional. If neither of them is present in the environment, I-Spy will use **\$ISPY_DIR/etc/sqlhosts** as the **sqlhosts** file. The order of precedence then, when looking for an **sqlhosts** file, is as follows:

1. If **INFORMIXSQLHOSTS** is present in the environment, this path is used.
2. If **INFORMIXDIR** is present in the environment, **\$INFORMIXDIR/etc/sqlhosts** is used.
3. If neither of the previous two variables are present in the environment, **\$ISPY_DIR/etc/sqlhosts** is used.

If you installed I-Spy on a computer that had other Informix products already installed, you already have at least **INFORMIXDIR** set in your environment and should not need to make any change. On the other hand, if you have installed I-Spy on a computer without other Informix products, you might want to set **INFORMIXSQLHOSTS** to the full path of the **sqlhosts** file you want to use or simply use the default of **\$ISPY_DIR/etc/sqlhosts**.

Shared-Library Search Path

You need to set the appropriate environment variable for shared-library searching to include **\$ISPY_DIR/lib**. If you do not have this setting in your environment, you will probably see an error similar to the following one when you run **ispy**.

```
ld.so.1: ./ispy: fatal: libifsql.so: can't open file: errno=2
Killed
```

Check with your system administrator to determine the correct environment variable to use for your platform.

- **SOLARIS** systems use **LD_LIBRARY_PATH** to specify directories to search for shared libraries. For example, if I-Spy is installed in **/usr/isp**:

```
$ LD_LIBRARY_PATH=/usr/isp/lib:$LD_LIBRARY_PATH
$ export LD_LIBRARY_PATH
```

- **HP-UX** systems use **SHLIB_PATH** to specify directories to search for shared libraries. For example, if I-Spy is installed in **/usr/isp**:

```
$ SHLIB_PATH=/usr/isp/lib:$SHLIB_PATH
$ export SHLIB_PATH
```

- **AIX** systems use **LIBPATH** to specify directories to search for shared libraries. For example, if I-Spy is installed in **/usr/isp**:

```
$ LIBPATH=/usr/isp/lib:$LIBPATH
$ export LIBPATH
```

It is important to place **\$ISPY_DIR/lib** in the front of the shared-library path list when you use I-Spy so that the appropriate shared libraries are found. If you have other Informix products installed on the same computer as I-Spy, you might already have the shared-library path setting configured for these products. In this case, it is still necessary to add the I-Spy **lib** directory to the front of the path list.

PATH

The final environment variable setting is to include **\$ISPY_DIR/bin** in your existing **PATH** setting. This allows you to execute the I-Spy programs while they are located in any directory on your computer. For example,

```
$ PATH=$ISPY_DIR/bin:$PATH
$ export PATH
```

Prepare the ISCONFIG Configuration File

The configuration file for I-Spy controls all aspects of how the product operates, where SQL activity is stored, what Informix database server names to use, rules that control client activity, and so on. The **\$ISPY_DIR/etc/isconfig.std** file resembles the following table.

shadowserver	on71_net	# Actual Informix Server
ispyserver	on71_spy	# Informix server value for clients
explaintags	0	# 0=off 1=on
logname	/usr/ispy/log/ispy.log	
activity_capture	0	# 0 = None 1 = Database 2 = File
activity_dbname	ispy	
activity_informixserver	on71_shm	
activity_dir	/usr/ispy/activity	
activity_id	0	
phase connect		
phase dlopen		
phase active		
phase disconnect		

To prepare the ISCONFIG file using a standard text editor

1. Make a copy of the **\$ISPY_DIR/etc/isconfig.std** file.
Store the new file in the **\$ISPY_DIR/etc** directory. Do not modify **isconfig.std**. Informix suggests that you choose a filename that reflects the name of the database server you want to log activity for. For example, **isconfig.sales_net** if your database server was named **sales_net**. Set your **ISPY_CONFIGFILE** environment variable to the name of your new file.

2. Edit your new **ISCONFIG** file to modify the configuration parameters that you have decided to change. For the initial configuration of I-Spy, you can leave most of the parameters set to their initial settings. The following parameters must be reviewed and changed if necessary:

- SHADOWSERVER
- ISPYSERVER
- LOGNAME
- ACTIVITY_DIR
- ACTIVITY_ID

For more information on the parameters, see [Chapter 3](#).

Prepare the sqlhosts File

The **sqlhosts** file contains information that is required to allow an Informix client application to connect to an Informix database server. For more information on the content of the **sqlhosts** file, see your *Administrator's Guide*.

If you installed I-Spy on a computer that already contained Informix products, you should already have an existing **sqlhosts** file. In this case, you must add a new line to that file that contains the ISPYSERVER name from the configuration file. Otherwise, you will have to create a new **sqlhosts** file that contains a line for both the SHADOWSERVER name and the ISPYSERVER name from the configuration file.

In either case, when you add the ISPYSERVER name to the **sqlhosts** file, you must choose an unused TCP port for I-Spy to listen on for client connections. This port must not be used by any other software installed on the computer. On most operating systems, TCP port numbers below a certain value, such as 1024, can only be accessed by user **root**. Since the I-Spy daemon process is started as user **informix**, choose a port number larger than this value. Check with your system administrator to determine an appropriate port number to use.

Start the ISPY Daemon

Start the I-Spy daemon process named **ispy** as user **informix**. Assuming you have set the appropriate environment variables and prepared the configuration file as described previously, you can start the daemon with the command **ispy** as follows:

```
$ ispy
$
```

The **ispy** process will now run in the background as a daemon listening for new client connections.

To verify that the daemon process is indeed running, you can execute **ispy** again with a **-r** command-line option. This command returns the process id number of the daemon if it is running or return no output if it is not running, as the following example shows:

```
$ ispy -r
358
$
```

Configuring Clients for I-Spy

To configure Informix client applications to use I-Spy, follow these steps:

1. Change the **INFORMIXSERVER** environment variable to the value of ISPYSERVER in the configuration file.
2. If the client is on a different computer than the I-Spy installation, add an entry in the client **sqlhosts** file for the ISPYSERVER name.

Important: It is possible to configure I-Spy so that no changes are required for client applications. For more information on how this is accomplished, see [“Advanced Configuration Options” on page 3-30](#).

Configuring INFORMIX-Connect

Use the Setnet32 utility to configure INFORMIX-Connect clients. The **Server Information** tab specifies the location of the database server, as Figure 2-1 shows.

Figure 2-1
*Specifying location
of database server
in Setnet32 utility*



In this example, the ISPYSERVER value is **sales_spy** and the **sqlhosts** entry for this server name is as follows

```
sales_spy      onsoctcp      cougar      6003
```

What to Do Next?

Depending on your environment, you might want to explore the following topics in more depth:

- To configure the web interface for I-Spy, see [Chapter 4](#).
- To configure activity logging, see [Chapter 6](#) on the activity database and the following configuration-file parameters in [Chapter 3](#).
 - ❑ ACTIVITY_CAPTURE
 - ❑ ACTIVITY_DBNAME
 - ❑ ACTIVITY_DIR
 - ❑ ACTIVITY_INFORMIXSERVER
- For more information on capturing the explain text for SQL statements, read about the **isexplain** utility on [page 5-3](#) and review the configuration-file parameter EXPLAINTAGS on [page 3-6](#).
- For advanced configuration options, such as using I-Spy with more than one Informix database server, see [Chapter 3](#).

Configuration File

General Configuration-File Syntax	3-3
Rules Overview	3-4
Rule Execution Order	3-6
Regular-Expression Syntax	3-6
ACTION	3-8
ACTION ENDESS	3-9
ACTION ERROR	3-9
ACTION EXECSQL	3-10
ACTION EXECUTE	3-10
ACTION EXPLAIN TAGS	3-12
ACTION NOLOG	3-13
ACTIVITY_CAPTURE	3-14
ACTIVITY_DBNAME	3-15
ACTIVITY_DIR	3-15
ACTIVITY_ID	3-16
ACTIVITY_INFORMIXSERVER	3-16
DEBUGFILE	3-17
DEBUGLEVEL	3-17
ECOST	3-18
EROWS	3-18
EXPLAIN TAGS	3-19

HOST	3-20
ISPYSERVER	3-21
ITIME	3-22
LOGFILE	3-22
PHASE	3-23
QTIME	3-23
ROWS	3-25
RULE	3-25
SHADOWSERVER	3-27
SQL	3-27
SQLLENGTH	3-28
SQLTRIM	3-28
USER.	3-30
Advanced Configuration Options	3-30
Monitoring Multiple Database Servers	3-31
Limitations	3-31
Configuring I-Spy with No Changes in the Client Environment	3-32
Multiple I-Spy Configurations, One Activity Database?	3-33

In This Chapter

This chapter discusses the I-Spy configuration file in detail. The configuration file defines all aspects of how I-Spy operates.

This chapter covers the following topics:

- General configuration-file syntax
- An overview of the rules system
- Regular-expression syntax
- Detailed syntax and descriptions for all configuration-file parameters.

General Configuration-File Syntax

The I-Spy configuration file consists of multiple lines. Each line contains a specific parameter. Blank lines are ignored, and anything after the # character on a line is ignored as a comment. A sample configuration file might resemble the following example:

SHADOWSERVER	godzilla_net	
ISPYSERVER	godzilla_spy	
EXPLAINTAGS	0	# 0=Off, 1=On
LOGNAME	/usr/isy/log/isy.log	
ACTIVITY_CAPTURE	2	# 0=None, 1=Database, 2=File
ACTIVITY_DBNAME	isy	

(1 of 2)

ACTIVITY_INFORMIXSERVER	ispy_net
ACTIVITY_DIR	/usr/ispy/activity
ACTIVITY_ID	1

PHASE CONNECT

PHASE DBOPEN

PHASE ACTIVE

RULE 1

USER informix

ITIME 1800

ACTION endsess

PHASE DISCONNECT

(2 of 2)

The order of the parameters in the configuration file is not significant, although it is common to place the rules section at the end of the file after all the general parameters.

All parameters such as **ispyserver** or **activity_capture** can be in lowercase or uppercase characters.

Rules Overview

The rules section of the configuration file controls various aspects of client connections. I-Spy separates each client session into four distinct phases: connection, database open, active, and disconnect. You can define rules in the configuration file for each phase. The phase parameter signifies that all rules in the file after that point are associated with that phase until either the end of the file, or another phase parameter is seen.

Consider the following section of a configuration file:

```

PHASE connect

RULE 1
USER informix
ACTION execute /usr/local/bin/ispay_connect.sh

PHASE dbopen

PHASE active

RULE 2
QTIME 300
ACTION error Query exceeded maximum execution time of 5
minutes.

PHASE disconnect

```

Rule 1 is only applied when a new client connection is made. Rule 2 is only applied in the active phase, or after a database is opened and the client is able to submit SQL statements.

Each rule contains a number of conditions and actions with each condition or action listed on a separate line in the file. If all the conditions for a rule are true, I-Spy executes all actions defined for that rule. Consider the following sample rule:

```

RULE 10
USER informix
HOST cheetah
ITIME 600
ACTION endsess
ACTION execute /usr/local/bin/ispay_endsess.sh

```

The USER, HOST, and ITIME parameters are all conditions for rule 10. If a client is user **informix**, the client is on a host named **cheetah**, and they are inactive for more than 10 minutes (ITIME 600), I-Spy performs two actions. The first action is to end the session, and the second is to execute the specified shell script.

Tip: There is no parameter that signifies the end of a rule. All conditions and actions that appear after a RULE parameter are associated with that rule until either another RULE or PHASE parameter is found, or the end of the file is reached.



Rule Execution Order

It is possible to have more than one rule match the execution of a single SQL statement. In this case, rules are executed in the order in which they appear in the configuration file, not in the order of the rule number.

Along the same lines, if a rule defines multiple actions to execute, they will be executed in the order in which they appear in the configuration file.

Consider the following rules:

```
PHASE active
RULE 1
SQLLENGTH 128
ACTION execute /usr/local/bin/long_sql.sh

RULE 2
ROWS 1000
ACTION execute /usr/local/bin/rows1000.sh
ACTION error Maximum 1000 rows exceeded
```

If an SQL statement longer than 128 characters is submitted, and it returns more than 1000 rows, the following actions occur in the order listed:

1. The script **/usr/local/bin/long_sql.sh** is executed.
2. The script **/usr/local/bin/rows1000.sh** is executed.
3. The statement is terminated with the error -746 I-Spy (2) Maximum 1000 rows exceeded.

Regular-Expression Syntax

Regular expressions provide a mechanism to find a specific string within another string. The four parameters, HOST, USER, SQL, and SQLTRIM, in the **isconfig** configuration file allow you to use regular expressions for matching. The ability to use regular expressions enables you to create more general rules than would be otherwise possible.

A regular expression (RE) is considered true if the characters within the expression find a match in the corresponding comparison element. Consider the case where the client submits the SQL statement “select * from customer”.



Tip: All the examples presented here will be contained within double quotes to signify the beginning and the end of the string, but the double quotes are not considered part of the string.

The RE “select” is true because the word *select* appears in the SQL statement, while the RE “SELECT” is not true, because the uppercase word SELECT does not appear.

The following table describes a number of characters that have special meanings when they are used in regular expressions.

Character	Description
RE*	A RE of a single character followed by an asterisk matches zero or more occurrences of the RE. For example, in the SQL statement “insert into tab values (1000)”, the RE “0*” matches the string “000”. If the asterisk is the first character in a RE, it has no special meaning and is treated as itself.
.	A period in a RE matches any single character. For example, in the SQL statement “select col1, col2 from customer”, the RE “col.” matches the strings “col1”, and “col2”.
^	If used as the first character of a RE, the circumflex matches the beginning of the string. For example, in the SQL statement “select * from customer”, the RE “^select” matches the string “select”. On the other hand, the same RE, “^select”, does not match the SQL statement “select * from customer” because the SQL statement has a blank character preceding the word select.

(1 of 2)

Character	Description
\$	If used as the last character of a RE, the dollar sign matches the end of the string. For example, in the SQL statement “delete from orders”, the RE “orders\$” matches the string “orders”, but does not match the SQL statement “delete from orders;”
[]	A string of characters enclosed in square brackets ([]) matches any single character in that string. If the first character after the left square bracket is a circumflex, ^, it matches any character except those listed after the circumflex.
-	The minus character (-) represents a range of characters. For example [0-9] is short for [0123456789]. For example, in the SQL statement “select * from customer1”, the RE “customer[0-9]” matches the string “customer1”, but does not match the SQL statement “select * from customer”.

(2 of 2)

For more information on regular expressions, see your system man pages on **regexp**.

ACTION

Syntax	ACTION	{ action keyword }
Example	ACTION	ENDSESS Session Terminated!
Required	No	

ACTION specifies a task for I-Spy to perform if all the conditions for a rule are true. Multiple actions might be defined for a rule, and they are executed in the order in which they appear in the configuration file. ACTION is associated with whatever rule immediately precedes it in the configuration file.

The following action keywords are available:

- ENDSSESSION
- ERROR
- EXECSQL
- EXECUTE
- EXPLAIN TAGS
- NOLOG

ACTION ENDSSESSION

Syntax	ACTION ENDSSESSION { client message }
Example	ACTION ENDSSESSION Session terminated!

ENDSESSION ends a client session and returns an error message to the client. The text on the line after the ENDSSESSION keyword is returned to the client application as an error message with a -746 error number. The maximum length of the error message returned to the client is 57 characters. Longer messages are truncated.

The **ispy** process handling the client session closes the database connection and exits after sending the error message to the client application. To continue working, the client must create a new database connection.

ACTION ERROR

Syntax	ACTION ERROR { client message }
Example	ACTION ERROR Statement canceled!

ERROR terminates an executing SQL statement and returns an error message to the client. All text on the line after the ERROR keyword is returned to the client application as an error message with a -746 error number. The maximum length of the error message returned to the client is 57 characters. Longer messages are truncated.

This action affects only the currently submitted or executing SQL statement. The client is able to continue submitting subsequent SQL statements after receiving the -746 error.

ACTION EXECSQL

Syntax	ACTION EXECSQL { SQL statement }
Example	ACTION EXECSQL set isolation to dirty read

EXECSQL submits SQL statements to the database server independent of the client application. The client application is unaware that the SQL submission has taken place. All text on the line after the EXECSQL keyword is sent to the database server as a single SQL statement.

This action is typically used in the **dbopen** phase of a session to correctly configure a client database environment. Typical SQL statements used are SET ISOLATION, SET LOCK MODE, SET ROLE, and EXECUTE PROCEDURE. For example, the following rule could be used for user **smith**:

```
phase dbopen
rule 1
user smith
action execsql set isolation to dirty read
action execsql set lock mode to wait 60
```



Warning: SQL statements submitted with this action are not allowed to return a result set because the **ispy** process would be unable to process it. If such a statement is submitted, it will be ignored and not executed by the database server.

ACTION EXECUTE

Syntax	ACTION EXECUTE	{ program path }
Example	ACTION EXECUTE	/usr/ispy/bin/ispymail.sh

EXECUTE runs a program on the computer where I-Spy resides. Supply the full path to the program to eliminate problems with relative references.

I-Spy forks itself to execute the program but does not wait for the program to complete. The following table lists the variables present in the environment for the program executed.

Variable	Description	Example
ISPY_CLIENHOST	The host name of the client computer	ISPY_CLIENHOST=cougar
ISPY_CLIENTIP	The IP address of the client computer	ISPY_CLIENTIP=192.147.100.15
ISPY_DBNAME	The name of the currently open database	ISPY_DBNAME=stores7
ISPY_ECOST	The estimated cost of the current SQL statement	ISPY_ECOST=3245
ISPY_EROWS	The estimated number of rows returned from the current SQL statement	ISPY_EROWS=45
ISPY_EXECUTE	The name of the program supplied for the EXECUTE keyword in the configuration file	ISPY_EXECUTE=/usr/isy/local/send mail.sh
ISPY_NFETCH	The number of fetch requests this client has sent to the database server for the current SQL statement	ISPY_NFETCH=3
ISPY_PREPARE	The date and time the current SQL statement was prepared	ISPY_PREPARE=1998-12-24 12:56:57
ISPY_ROWS	The number of rows returned to the client for the current SQL statement	ISPY_ROWS=5000
ISPY_RULE	The number of the rule that caused this execute action to start	ISPY_RULE=10
ISPY_SQL	The text of the current SQL statement	ISPY_SQL=select count(*) from customer
ISPY_START	The date and time the current SQL statement began executing	ISPY_START=1998-12-24 12:56:58
ISPY_USERNAME	The username for the client connection	ISPY_USERNAME=jsmith

The following sample shell script named **\$ISPY_DIR/BIN/ISPYMAIL.SH** demonstrates how you can use these environment variables to send an email:

```
#!/bin/sh

MAILTO="dba@company.com"
SUBJECT="I-Spy Alert"

echo "

This email is being generated from the script
$ISPY_EXECUTE.

USERNAME      $ISPY_USERNAME
CLIENTHOST    $ISPY_CLIENTHOST
CLIENTIP      $ISPY_CLIENTIP
DBNAME        $ISPY_DBNAME
RULE          $ISPY_RULE
SQL           $ISPY_SQL
ECOST         $ISPY_ECOST
EROWS         $ISPY_EROWS
ROWS          $ISPY_ROWS
NFETCH        $ISPY_NFETCH
START         $ISPY_START
PREPARE       $ISPY_PREPARE

" | mailx -s "$SUBJECT" $MAILTO
```

ACTION EXPLAINTAGS

Syntax	ACTION EXPLAINTAGS	{ 0 1 }
Example	ACTION EXPLAINTAGS	0

EXPLAINTAGS specifies whether or not to append a unique comment identifier to each SQL statement submitted by clients. The **isexplain** utility uses this unique identifier to load explain text into the activity database and link it to the original SQL statement. The values 0 and 1 disable and enable this feature, respectively.

EXPLAIN TAGS is also used as a global parameter for the configuration file instead of an action keyword as shown here. This allows you to have a global setting for the value but also change the value for specific sessions.

ACTION NOLOG

Syntax	ACTION NOLOG
Example	ACTION NOLOG

NOLOG specifies that this specific SQL statement should not be saved in the activity database. A common use for this action is to filter out those SQL statements that you are not interested in saving. For example, you could use the following rules to filter out all INSERT, UPDATE, and DELETE SQL statements as well as any that reference the system catalog tables **systables**, **syscolumns**, and **sysindexes**.

```
rule 1
SQL "insert"
ACTION NOLOG
rule 2
SQL "update"
ACTION NOLOG
rule 3
SQL "delete"
ACTION NOLOG
rule 4
SQL "systables"
ACTION NOLOG
rule 5
SQL "syscolumns"
ACTION NOLOG
rule 6
SQL "sysindexes"
ACTION NOLOG
```

ACTIVITY_CAPTURE

Syntax	ACTIVITY_CAPTURE { 0 1 2 }
Example	ACTIVITY_CAPTURE 1
Required	Yes

ACTIVITY_CAPTURE specifies what the I-Spy daemon should do with the SQL activity that is captured for a client connection. You must specify one of the following values:

- 0 - No capture
- 1 - Save SQL statements in the activity database
- 2 - Save SQL statements in the file system for later loading into the activity database

ACTIVITY_CAPTURE is a global parameter and is in effect for all clients that connect to I-Spy. You can use the NOLOG action of a rule to disable the logging of a specific SQL statement if the ACTIVITY_CAPTURE value is set to either 1 or 2.



Important: If the *ispy* daemon encounters any type of database error while inserting rows into the activity database, it will close that connection and begin logging SQL statements to the file system. The daemon writes a message similar to the following example into the log file indicating that this action has taken place:

```
SQLCODE 9999 insert( ispy_session ) line 345, filesystem
capture enabled
```

This behavior is a safety net to ensure that SQL activity from clients is always logged.

ACTIVITY_DBNAME

Syntax	ACTIVITY_DBNAME	{ dbname }
Example	ACTIVITY_DBNAME	ispy
Required	Yes	

ACTIVITY_DBNAME specifies the name of the database used to record client activity. [Chapter 6](#) contains more information on creating and configuring the activity database. While you can use any database name you like to record client activity for I-Spy, it is recommended that you use the name **ispy** to avoid confusion.

ACTIVITY_DIR

Syntax	ACTIVITY_DIR	{ directory path }
Example	ACTIVITY_DIR	/usr/ispy/activity
Required	Yes	

ACTIVITY_DIR specifies a full path to a directory used for files when ACTIVITY_CAPTURE is set to 2.

Each client connection to I-Spy creates two files in this directory. The first file is named **session.#**, where # is a unique number that I-Spy generates to uniquely identify each session. The second file is named **activity.#**, where # is the same unique number used for the session file. Each file contains information in standard Informix load format. You can use the **isload** utility to load the activity database with any files that exist in this directory.

Important: Monitor the contents of this directory even if you have ACTIVITY_CAPTURE set to 1. If I-Spy encounters any type of database error while it saves information to the activity database, it writes an error message into the log file and switches to saving information in the file system. This allows clients to work without being interrupted and still record all of their activity.



ACTIVITY_ID

Syntax	ACTIVITY_ID	{ integer }
Example	ACTIVITY_ID	5
Required	Yes	

ACTIVITY_ID uniquely identifies multiple I-Spy instances that log into the same activity database. The integer value specified is used to create the value generated for the **act_explain_id** column of the **ispy_activity** table.

Each I-Spy instance that logs to the same activity database should have a unique value for ACTIVITY_ID. Failure to define unique values for ACTIVITY_ID results in insert errors when you log into the activity database, or when you use the **isload** utility after logging to the file system has taken place.

ACTIVITY_INFORMIXSERVER

Syntax	ACTIVITY_INFORMIXSERVER	{ dbserver name }
Example	ACTIVITY_INFORMIXSERVER	menlo_net
Required	Only if ACTIVITY_CAPTURE = 1	

ACTIVITY_INFORMIXSERVER specifies the Informix database server that contains the database that ACTIVITY_DBNAME specifies. The I-Spy daemon uses that database at the specified database server to save all client activity.

The server name used for ACTIVITY_INFORMIXSERVER must be present in the **sqlhosts** file that I-Spy uses.

DEBUGFILE

Syntax	DEBUGFILE	{ file path }
Example	DEBUGFILE	/usr/ispay/log/debug.log
Required	No	

DEBUGFILE specifies the full path to a file that is used for writing debug information from the **ispay** daemon process. The debug file is intended for use by Informix Technical Support, and the format and contents of this file are likely to change.

DEBUGLEVEL

Syntax	DEBUGLEVEL	{ integer }
Example	DEBUGLEVEL	5
Required	No	

DEBUGLEVEL is an integer value that specifies the amount of debugging information to place in the debug file. It defaults to a value of 0, which indicates no debugging and has a maximum value of 300. Values above 5 tend to produce a considerable amount of debug information.

DEBUGLEVEL greater than 0 should only be used in cases where you want to verify that I-Spy is working, or if instructed by Informix Technical Support. Debugging introduces a considerable amount of overhead for a client connection and should not be used for production systems.

ECOST

Syntax	ECOST	{ integer }
Example	ECOST	9999
Required	No	

ECOST is a rule condition that specifies the estimated cost of an SQL statement submitted for execution by a client application. If the estimated cost of the statement is equal to or greater than the value supplied, the condition is considered true. ECOST is associated with whatever rule immediately preceded it in the configuration file.

EROWS

Syntax	EROWS	{ integer }
Example	EROWS	9999
Required	No	

EROWS is a rule condition that specifies the estimated number of rows in the result set of an SQL statement submitted for execution by a client application. If the estimated number of rows for the result set of the statement is equal to or greater than the value supplied, the condition is considered true. ROWS is associated with whatever rule immediately preceded it in the configuration file.

EXPLAINTAGS

Syntax	EXPLAINTAGS	{ 0 1 }
Example	EXPLAINTAGS	0
Required	Yes	

EXPLAINTAGS specifies whether or not to append a unique comment identifier to each SQL statement submitted by clients. The **isexplain** utility uses this unique identifier to load explain text into the activity database and link it to the original SQL statement. The values 0 and 1 disable and enable this feature, respectively.

If a client submitted the following SQL statement:

```
select * from customer ;
```

and EXPLAINTAGS was set to 1, I-Spy would change the SQL statement as follows before sending it to the shadow database server:

```
select * from customer ;  
--ISPY0-3-56
```

The database server would ignore the appended text since it is formatted as a comment, but it would be written to the **sqexplain.out** file for the **isexplain** utility to use later.

This feature is configurable because appending the comment does add additional overhead to client connections that you might want to avoid. You can also specify EXPLAINTAGS as a rule action if you only want to add this functionality in certain situations. For example, if you only want to use **isexplain** to capture explain text for users **john** and **karen**, you would set EXPLAINTAGS to 0 and use the following two rules:

```
EXPLAINTAGS 0

PHASE connect

RULE 1
USER john
ACTION EXPLAINTAGS 1

RULE 2
USER karen
ACTION EXPLAINTAGS 1
```

HOST

Syntax	HOST	{ hostname }
Example	HOST	server1
Required	No	

HOST is a rule condition and specifies a host name to match against the client host name. If the names match, the condition is considered true. HOST is associated with whatever rule immediately preceded it in the configuration file.

Regular expressions are permitted for the host name, allowing you flexibility in matching hosts. For more information on regular expressions, see [“Regular-Expression Syntax” on page 3-6](#).

ISPYSERVER

Syntax	ISPYSERVER	{ servername }
Example	ISPYSERVER	cougar_spy
Required	Yes	

ISPYSERVER defines a new server name for client connections. Connections to this new server name are routed to the database server defined by the SHADOWSERVER parameter. For example, consider the following **sqlhosts** file:

```
cougar_net ontlitcp cougar port1
cougar_spy ontlitcp cougar port2
```

The server name **cougar_net** is the direct connection to the actual Informix database server, or shadow server, and the **cougar_spy** entry is the new entry for I-Spy. The ISPYSERVER parameter should be set to **cougar_spy**. The I-Spy daemon process will listen on the port specified (port2) for client connections just as the database server would. When a new client connection is received, the daemon process will open a connection to the shadow server, **cougar_net** in this case, and route all message traffic to this database server.

Set **INFORMIXSERVER** environment variable to the value of ISPYSERVER for clients that you want to use I-Spy.

Warning: The ISPYSERVER name must be exactly the same length as the SHADOWSERVER name.



ITIME

Syntax	ITIME	{ seconds }
Example	ITIME	3600
Required	No	

ITIME IS a rule condition that specifies the amount of inactivity time for a client connection in seconds. Another way to say this is the amount of time a client has not submitted a request to the database server. If the amount of time is greater than the value supplied, the condition is considered true. ITIME is associated with whatever rule immediately preceded it in the configuration file.

You can only combine the ITIME condition with the USER and HOST conditions within a rule.

LOGFILE

Syntax	LOGFILE	{ logfile path }
Example	LOGFILE	/usr/ispay/log/ispay.log
Required	Yes	

LOGFILE specifies a file used to record messages from the I-Spy daemon process. The daemon writes messages to this file when it starts and terminates and if any type of warning or fatal error condition is encountered. Each line of the file contains the date and time the message was written, for example:

```
10-30 15:14:29 I-Spy beginning execution, pid 27433
```

The command **ispay -m** displays the last 20 lines of the log file.

PHASE

Syntax	PHASE	{ connect dbopen active disconnect }
Example	PHASE	dbopen
Required	No	

I-Spy separates each client connection into one of four distinct phases, connection, database open, active, or disconnect. The PHASE parameter indicates which phase should be associated with the rules that follow in the configuration file. The phase specified is in effect until either another PHASE parameter is seen, or the end of the file is reached.

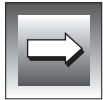
It is not required to specify all phases in the configuration file, only those phases in which you want to define rules.

Rules in the connect, dbopen, and disconnect phase can only specify the USER and HOST conditions: ITIME, QTIME, ROWS, SQL, and SQLTRIM are not allowed.

QTIME

Syntax	QTIME	{ seconds }
Example	QTIME	300
Required	No	

QTIME is a rule condition that specifies the amount of query-processing time in seconds for a single SQL statement. Another way to say this is the amount of time the database server spends processing all client requests for the SQL statement. If the amount of time is greater than the value supplied, the condition is considered true. QTIME is associated with whatever rule immediately preceded it in the configuration file.



You can only combine the QTIME condition with the USER, HOST, SQL, or SQLTRIM conditions within a rule.

Important: *It is important to understand how I-Spy calculates the query time for an SQL statement because it might not correspond exactly to your assumption. Each client request to the database for an SQL statement causes query time to accumulate until the database server responds to the request. Time between client requests but before the query is completed is not considered part of query time.*

Consider the following example:

A client declares a cursor for an SQL statement that will return 1000 rows from the database server. The client then opens the cursor and fetches the first 50 rows. The time required for the database server to open the cursor and return the first 50 rows is accumulated in query time for this SQL statement. At this point, the client performs some other type of processing that makes no further requests of rows for this SQL statement from the database server. This time is not accumulated in query time, although the SQL statement is still *executing* because the cursor is open.

Assume for this example that a rule is defined that would match this SQL statement and QTIME is set to 300 seconds or 5 minutes. If the client spends more than 5 minutes performing this other processing, the QTIME condition *would not become true* and the actions would not be triggered.

Later the client fetches the remaining rows of the result set and closes the cursor. These operations would be accumulated in query time for the statement. Only if the accumulated time for all fetch requests to the database exceeded 5 minutes would the QTIME condition become true.

ROWS

Syntax	ROWS	{ integer }
Example	ROWS	1000
Required	No	

ROWS is a rule condition that specifies the number of rows in the result set for an SQL statement. If the number of returned rows is greater than the value supplied, the condition is considered true. ROWS is associated with whatever rule immediately preceded it in the configuration file.

You can only combine the ROWS condition with the USER, HOST, SQL, or SQLTRIM conditions within a rule.

The database server returns rows to the client in groups, rather than one at a time. This results in queries not being stopped at exactly a specific number of rows, but within one grouping of rows. For example, if the database server is returning 23 rows at a time, the 1st group will contain rows 1-23, the 2nd rows 24-46, on to the 43rd group containing rows 967-989, and the 44th containing 990-1012. With a rows condition of 1000, I-Spy will prevent further rows from returning, but the client will see all rows up to and including 1012.

RULE

Syntax	RULE	{ number }
Example	RULE	39
Required	No	

RULE specifies the start of a new rule that is applied to each client connection. Rules can have any number of conditions and actions associated with them that appear on the lines that follow the RULE parameter.

Each rule in the configuration file must be assigned a unique number. This number is used in the ERROR action and the **sess_abortrule**, or **act_abortrule** columns of the activity database. This provides you with the ability to reference the exact rule that caused a specific action to be taken.

If all actions defined for a rule are true, then all defined actions for that rule are executed. The following table lists all conditions available for a rule.

Condition Parameter	Description
ECOST	The estimated cost for the execution of the SQL statement
EROWS	The estimated number of rows in the result set for the SQL statement
HOST	The hostname of the client.
ITIME	Inactivity time. The amount of time that has elapsed without the client submitting a request to the database server.
QTIME	Query time. The amount of execution time elapsed for a single SQL statement.
ROWS	The number of rows returned for an SQL statement
SQL	The text of an SQL statement
SQLLENGTH	The length of an SQL statement
SQLTRIM	The text of an SQL statement translated for easier expression matching
USER	The username of the client

SHADOWSERVER

Syntax	SHADOWSERVER	{ INFORMIXSERVER }
Example	SHADOWSERVER	cougar_net
Required	Yes	

This parameter specifies the Informix server name of the database server that clients connect to through I-Spy. If you already have users connecting to an existing database server, you would set SHADOWSERVER to the current value of INFORMIXSERVER. For example, if you had the following entry in your **sqlhosts** file:

```
cougar_net ontli tcp cougar port1
```

you would set SHADOWSERVER to the value **cougar_net**. The server name you use must be a network connection. I-Spy does not support any other connection type.

SQL

Syntax	SQL	“expression”
Example	SQL	“systables “
Required	No	

SQL is a rule condition that specifies an SQL statement to compare with. If the value supplied matches the SQL statement submitted by the client, the condition is considered true. SQL is associated with whatever rule immediately preceded it in the configuration file.

You cannot combine the SQL condition with either the QTIME or ROWS condition within a single rule.

Regular expressions are permitted for SQL, allowing you flexibility in matching statements. For more information on regular expressions see [“Regular-Expression Syntax” on page 3-6](#).

The value supplied must be contained with double quotes to delineate the begin and end of the expression. A `\` sequence can be used to include a double quote character as part of the expression.

SQLLENGTH

Syntax	SQLLENGTH	{ integer }
Example	SQLLENGTH	256
Required	No	

SQLLENGTH is a rule condition that specifies the length of an SQL statement submitted for execution by a client application. If the length of the statement is equal to or greater than the value supplied, the condition is considered true. SQLLENGTH is associated with whatever rule immediately preceded it in the configuration file.

SQLTRIM

Syntax	SQLTRIM	“expression”
Example	SQLTRIM	“systables “
Required	No	

SQLTRIM is a rule condition that specifies an SQL statement to compare with. If the value supplied matches the SQL statement submitted by the client, the condition is considered true. SQLTRIM is associated with whatever rule immediately preceded it in the configuration file.

You cannot combine the SQLTRIM condition with either the QTIME or ROWS condition within a single rule.

Regular expressions are permitted for SQLTRIM, allowing you flexibility in statements. For more information on regular expressions, see [“Regular-Expression Syntax” on page 3-6](#).

The value that you supply must be enclosed in double quotes to delineate the beginning and end of the expression. You can use a `\` sequence to include a double quote character as part of the expression.

SQLTRIM differs from the SQL condition in that it trims the client-supplied SQL statement into a simpler form before an expression match is attempted. The trim process consists of the following three steps:

1. All characters in the statement are converted to lowercase.
2. All leading and trailing white space is removed. White space is defined as blank, tab, and newline characters.
3. All remaining spans of more than one white-space character are reduced to a single blank character.

The process should make it easier to write a regular expression. Consider the following SQL statement submitted by the client:

```
SELECT
  *
from
  customer
where
  last_name = 'jones'
```

It would be difficult to create a regular expression that looks for any SELECT statements that include the **customer** table because of the embedded newline characters and the possibility that all SQL keywords could be in uppercase or lowercase. SQLTRIM makes the job simpler because it transforms the submitted SQL into this form before the expression match is attempted:

```
select * from customer where last_name = 'jones'
```

Now you could use SQLTRIM `“select.*from.*customer.*where”` to match this type of SELECT statement.

The SQLTRIM condition does not change in any way the actual text of the SQL statement that is sent to the database server. The transformation described earlier is only an internal process that the I-Spy daemon uses.



Tip: *SQLTRIM and SQL conditions are both supplied because there is more overhead in using SQLTRIM to transform each SQL statement before expression matching is performed. If you can create an expression that satisfies your needs using only the SQL condition, that is preferable. Use SQLTRIM only if you are having difficulty doing what you need using only SQL.*

USER

Syntax	USER	{ username }
Example	USER	johnr
Required	No	

USER is a rule condition that specifies a user name to match against the client's user name. If the names match, the condition is considered true. USER is associated with whatever rule immediately preceded it in the configuration file.

Regular expressions are permitted for the user name, allowing you flexibility in matching clients. For more information on regular expressions, see [“Regular-Expression Syntax” on page 3-6](#).

Advanced Configuration Options

This section discusses a number of advanced configuration options for I-Spy. It covers the following topics:

- Using I-Spy to monitor more than one Informix database server
- Configuring I-Spy with no changes to the client environment
- Using the same activity database with multiple I-Spy configurations

Monitoring Multiple Database Servers

Each **isconfig** configuration file is associated with the monitoring of a single Informix database server. To monitor more than one database server, you must create a separate configuration file for each one.

Use the `ISPY_CONFIGFILE` environment variable to specify which configuration file you are using.

The following parameters within each configuration file should be unique.

- ★ `SHADOWSERVER`
- ★ `ISPYSERVER`
- ★ `LOGFILE`
- ★ `ACTIVITY_DIR`

You must start an I-Spy daemon for each configuration file. Each configuration will have a separate server name, log file, and activity directory.

This setup works in exactly the same manner as multiple `onconfig` configuration files for multiple instances of an Informix database server on the same computer.

Limitations

Be aware of the following limitation when you configure multiple **isconfig** configuration files on the same computer. The default Web interface in **\$ISPY_DIR/web** only supports a single configuration.

The Web interface is restricted to one configuration because a single **web.cfg** file exists in the **\$ISPY_DIR/web** directory to define which **isconfig** configuration file to reference. One workaround to this issue would be to copy the entire Web directory to a new directory. Then update the **web.cfg** file in the new directory to reference the second **isconfig** configuration file. Finally, update your Apache Web server configuration to define a new alias that references the new directory structure.

Configuring I-Spy with No Changes in the Client Environment

It is possible to configure I-Spy so that no changes are required in the client environment for those clients to access I-Spy. This feature might be advantageous in an existing environment where updating client environments is difficult.

Another reason this configuration option might be useful is that it allows you to determine whether clients connect directly to the Informix database server, or to I-Spy. You can make this choice from the computer where the database server resides, instead of having to update the environment of each client.

For this process to work, I-Spy must be installed on the same computer as the shadow database server because the third column of the client **sqlhosts** file is the name of the host where the database server resides. If I-Spy was installed on a different computer from the one that contains the database server, you would always be required to update the client configuration to switch between I-Spy and a direct connection to the database server.

Consider the case where you have I-Spy installed on a computer that runs a database server currently named **sales_net**. Your various configuration files resemble:

Your **isconfig** configuration file resembles:

```
SHADOWSERVER    sales_net
ISPYSERVER      sales_spy
```

Your **sqlhosts** configuration file resembles:

```
sales_net    onsoctcp    server1    port1
sales_spy    onsoctcp    server1    port2
```

Your **onconfig** configuration file resembles:

```
INFORMIXSERVER sales_net
```

You should configure clients with the single **sqlhosts** entry of **sales_net** and set their **INFORMIXSERVER** environment variable to **sales_net**. In this way, based on the preceding configuration files, they will connect directly to the Informix database server and not use I-Spy at all.

To route clients through I-Spy without making any changes in the client configuration

1. Make sure all clients have disconnected from the database and then shut down both the database server and I-Spy
2. Swap the values of SHADOWSERVER and ISPYSERVER in the **isconfig** configuration file. The **isconfig** file should now resemble:

```
SHADOWSERVER    sales_spy
ISPYSERVER       sales_net
```

3. Change the value of DBSERVERNAME in **onconfig** to reference sales_spy instead of sales_net:

```
DBSERVERNAME    sales_spy
```



Tip: You might need to change DBSERVERALIAS instead of DBSERVERNAME if sales_net was originally defined using DBSERVERALIAS.

4. Start up the database server and I-Spy.

What you have done is swap the INFORMIXSERVER and ISPYSERVER names so that the database server is now listening on the name sales_spy, and I-Spy is listening on the name sales_net. Clients will now connect through I-Spy since they reference the name sales_net.

You can configure clients to connect directly to the database server by changing the configuration back to the original set of files.

Multiple I-Spy Configurations, One Activity Database?

Multiple I-Spy instances can all log into a single activity database as long as the ACTIVITY_ID value in the configuration file of each instance is unique. The ACTIVITY_ID value is used to populate the **act_explain_id** column of the activity table.

This feature is useful if you are running Informix Dynamic Server with Advanced Decision Support and Extended Parallel Options and have multiple connection coservers configured. Each connection coserver would be running an instance of I-Spy, but they would all log to a single activity database.

Configuring the Web Interface

Preparing the web.cfg Web Configuration File	4-4
Configuring Web Server Directory Access	4-5
Adding a URL Alias	4-5
Allowing CGI Execution and Directory Access	4-6
Granting Access to the Activity Database	4-7
Testing the Web Interface.	4-7
Configuration of Basic Web Security	4-8
Copy the Informix I-Spy User Manual	4-9
Custom Reports Link on the Reports Page.	4-10
Configuration for Netscape FastTrack Server	4-12
Create New Document Directory	4-12
Activate the CGI File Type	4-13

In This Chapter

This chapter describes how to configure the Web interface for I-Spy. The Web interface is an optional component of I-Spy. You can perform all administrative activities using the command-line utilities without the Web interface. The advantage to using the Web interface is that it eliminates the need for you to remember all the details of your I-Spy installation, such as configuration filenames, directory paths, and so on.

This chapter discusses the configuration of the Apache Web server, Version 1.2.5. You can directly apply the concepts presented to almost any other Web server, although the specific syntax will be different than that presented here. Contact your Web administrator for more information on your particular Web server. For more information on the Apache Web server, see <http://www.apache.org>.



Important: It is required that a Web server be installed on the same computer where I-Spy is located. It is not possible to use the Web interface if a Web server is not available on the same computer.

This chapter describes the following tasks:

- Preparing the Web configuration file **web.cfg**
- Configuring Web-server directory access for the **\$ISPY_DIR/web** directory
- Granting access to the activity database
- Testing the Web interface

The following tasks are optional and can be configured depending on your environment:

- Configuring basic Web security, if applicable
- Copying the *Informix I-Spy User Manual* into the **\$ISPY_DIR/web/userman** directory
- Custom reports link on the report page.

Preparing the web.cfg Web Configuration File

All files necessary for the Web interface are located in the **\$ISPY_DIR/web** directory. You must create a new configuration file within this directory named **web.cfg**. The **web.cfg** file will contain any environment variable settings needed for the interface.

To prepare the web.cfg file using a standard text editor

1. Copy the file **\$ISPY_DIR/web/web.cfg.std** to **\$ISPY_DIR/web/web.cfg**.

Make sure the file has an owner and group **informix** and a permission mode of 644. Do not modify the **web.cfg.std** file. All Common Gateway Interface (CGI) scripts in the Web interface look for a file named **web.cfg** in the Web directory. The file cannot have a different name.
2. Edit the **web.cfg** file, placing all necessary I-Spy environment variables on a separate line. Each line has the form **VARNAME=value**, as the following example shows:

```
ISPY_DIR=/usr/isy
```

The following environment variables must be present in this file:

- ❑ **ISPY_DIR**
- ❑ **ISPY_CONFIGFILE**
- ❑ Shared-library search path



Optionally, you might need to include **INFORMIXSQLHOSTS** if you are not using the default **sqlhosts** file **\$ISPY_DIR/etc/sqlhosts**.

The *shared-library search path* represents the appropriate value for your operating system. For example, on SOLARIS you would use **LD_LIBRARY_PATH**. This environment variable is discussed in [Chapter 2](#).

Important: You must include **ISPY_CONFIGFILE** even if you are using the default value of **\$ISPY_DIR/etc/isconfig**.

The **web.cfg** file should not include any lines other than those that set specific environment variables. Even blank lines are not allowed. For example, if you installed I-Spy in **/usr/isy** and were on a HP-UX computer, your **web.cfg** file would resemble the following example:

```
ISPY_DIR=/usr/isy
ISPY_CONFIGFILE=/usr/isy/etc/isconfig.server1
SHLIB_PATH=/usr/isy/lib
INFORMIXSQLHOSTS=/usr/isy/etc/sqlhosts.server1
```

Configuring Web Server Directory Access

Follow these steps to configure your Web server to access the **\$ISPY_DIR/web** directory:

1. Add a URL alias so that you can use the URL `http://hostname/isy/` to start the Web interface.
2. Allow directory access and CGI script execution for **\$ISPY_DIR/web**.

Adding a URL Alias

The Apache Web server allows you to define aliases for pathnames so that you do not have to enter the entire path to retrieve a file. Informix recommends that you configure an alias to simplify access to the I-Spy Web interface.

In the Apache **srm.conf** file, add the following line:

```
Alias /isy/ /usr/isy/web/
```



This example assumes that you have installed I-Spy in **/usr/ispay**. Replace this directory with your installation directory name, if appropriate.

This alias allows you to enter `http://hostname/ispay/` to bring up the first page of the Web interface.

Warning: *It is assumed that your Apache Web server is configured to automatically retrieve a file named **index.html** if one is not specified. This name of this file is identified by the `DirectoryIndex` setting in the **srm.conf** file. If your configuration specifies a different filename, you have two options: (1) you can change this setting to **index.html**, or (2) you could rename the file **\$ISPY_DIR/web/index.html** to the setting of `DirectoryIndex`. Informix recommends that you use the first option, because changing the name of **index.html** could lead to problems when you install future versions of I-Spy.*

Allowing CGI Execution and Directory Access

By default, most Apache server installations only allow files to be retrieved from specific directories identified by `<Directory>` sections in the **access.conf** configuration file. You need to create a `<Directory>` section for access to the files in **\$ISPY_DIR/web**.

Add the following lines to your Apache **access.conf** configuration file:

```
<Directory /usr/ispay/web>
Options Indexes FollowSymLinks ExecCGI
AllowOverride None
order allow,deny
allow from all
</Directory>
```

Change the path **/usr/ispay/web** to reflect where you have installed I-Spy.

Also verify that your Apache server has a handler defined to handle CGI scripts. Make sure that the following line is present in the **srm.conf** configuration file:

```
AddHandler cgi-script .cgi
```

Granting Access to the Activity Database

The report section of the Web interface requires access to the activity database. For more information on granting the Web server this access, see [“Grant Connect Privilege” on page 6-6](#).

Testing the Web Interface

At this point, you should be able to use the Web interface. Open the following URL in your Web browser, where **hostname** is the name of the computer where you have I-Spy installed:

```
http://hostname/isp/
```

You should see the following window.

If this page does not display, review the preceding configuration topics to make sure everything is configured correctly.

Configuration of Basic Web Security

The Web interface contains the following buttons on the left side of the display: **Status**, **Startup**, **Shutdown**, **Config File**, **Clients**, **Reports**, **View Log**, and **User Manual**. You might want to restrict access to one or more of these options. For example, you might want to allow some users access to the Reports section of the interface, but not the ability to start up or shut down the I-Spy daemon process.

You can use basic Web authentication to force a Web browser to prompt for a user name and password when any of the buttons is selected, thereby controlling access to the different options.

The Apache Web server allows you to configure basic authentication for a specific directory. Each of the buttons on the I-Spy Web interface is associated with a specific CGI script in a directory under **\$ISPY_DIR/web**, as the following table shows.

Button	Directory
Status	\$ISPY_DIR/web/status
Startup	\$ISPY_DIR/web/startup
Shutdown	\$ISPY_DIR/web/shutdown
Config File	\$ISPY_DIR/web/edit/editconfg
Clients	\$ISPY_DIR/web/clients
Reports	\$ISPY_DIR/web/reports
View Log	\$ISPY_DIR/web/viewlog
User Manual	\$ISPY_DIR/web/userman

To require a user name and password for a specific button, add a <Directory> section for the corresponding directory in the Apache **access.conf** configuration file. For example, to require a password for startup, add the following lines:

```
<Directory /home2/ispy/current/web/startup>
AuthType Basic
AuthName ispy
require user markj
AuthUserFile /usr/ispy/web/.htpasswd
AuthGroupFile /usr/ispy/web/.htgroup
```

You also must add a **.htpasswd** and **.htgroup** file to your **\$ISPY_DIR/web** directory. For the preceding example, the **.htgroup** file contains:

```
group1: markj
```

The **.htpasswd** file contains:

```
markj:XrtIiBSBpwZQA
```

For this configuration example, the Web browser will prompt for a user name and password and only accept **markj** as the user name and the password contained in the **.htpasswd** file. To create the **.htpasswd** file, use the **htpasswd** utility that is part of the Apache Web server distribution.

Tip: For more detailed information on configuring user authentication for the Apache Web server, see <http://www.apacheweek.com/features/userauth>.



Copy the Informix I-Spy User Manual

The **User Manual** button on the Web interface expects to find a file with a **.pdf** extension in the **\$ISPY_DIR/web/userman** directory. To make the user manual available from the Web interface, you need to copy the **.pdf** file from your installation media into this directory.

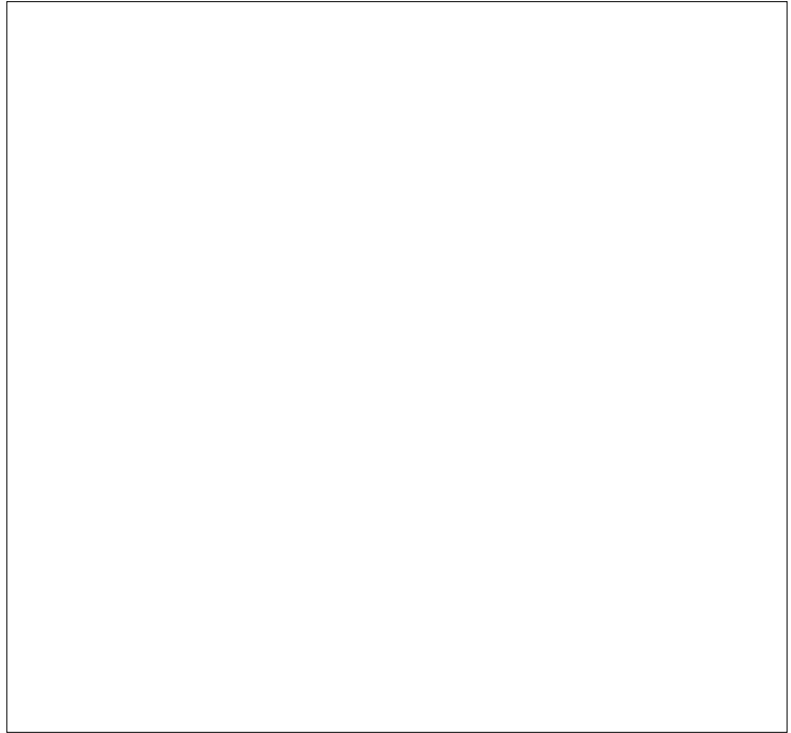
The exact name of the **.pdf** file is not important, because the Web interface automatically adjusts to whatever file in the **\$ISPY_DIR/web/userman** directory ends with a **.pdf** extension.

To view the user manual, the Adobe Acrobat reader must be installed on the computer where the Web browser resides.

Custom Reports Link on the Reports Page

The **Reports** page of the Web interface contains links to a number of predefined reports, as the following window shows.

If the file `$ISPY_DIR/web/reports/custom_reports.html` exists, a link to this file is shown on the **Reports** page as the following window shows.



The I-Spy distribution contains the file **web/reports/custom_reports.html.std** that you can copy to **custom_reports.html**.

You can use this file to create your own page of links that generate reports that will automatically be available with future releases of I-Spy.

Configuration for Netscape FastTrack Server

To configure the Netscape FastTrack server for the I-Spy Web interface, follow these steps:

1. Create an additional document directory.
2. Activate the CGI file type.

Create New Document Directory

To create a new document directory, follow these steps:

1. Click the **Content Management** button on the FastTrack administrative interface.
2. Select **Additional Document Directories** from the menu on the left.
3. Insert **/ispy** in the **URL prefix** text box.
4. Insert your **\$ISPY_DIR/web** directory in the **Map to Directory** text box.
5. Click **OK** to add the new directory.

The following window shows the result of adding a new document directory.

Activate the CGI File Type

To activate the CGI file type, follow these steps:

1. Click the **Programs** button on the FastTrack administrative interface.
2. Select **CGI File Type** from the menu on the left.
3. Select **Yes** for **Activate CGI as a file type?**.

The following window shows the result of making this change.

You should now be able use the URL `http://hostname/isp/` from your Web browser to access the I-Spy Web interface.

I-Spy Utilities

isexplain: Load Explain Text.	5-3
isload: Load Files into the Activity Database	5-5
ispy: I-Spy Daemon Process	5-6

In This Chapter

This chapter provides reference material for all of the I-Spy utilities. The utilities allow you to execute administrative tasks directly from your command line. The following utilities are documented in this chapter:

- **isexplain**
- **isload**
- **ispy**

isexplain: Load Explain Text

Use the **isexplain** utility to scan **sqexplain.out** files generated by Informix database servers for query plans that you can load into the I-Spy activity database.

Syntax

```
isexplain -c configfile -v explainfile
```

The diagram illustrates the syntax of the `isexplain` command. It shows the command name `isexplain` followed by two optional flags, `-c configfile` and `-v`, and a required argument `explainfile`. The flags and argument are enclosed in a box, indicating they are optional. The `-v` flag is shown with a long underline, suggesting it is a verbose flag.

Options

- c configfile** This option specifies the full path to the configuration file that you want to use. If a configuration file is not specified, **isexplain** looks at the environment variable **ISPY_CONFIGFILE** for the full path. If this environment variable is not set, **isexplain** uses the default directory **\$ISPY_DIR/ETC/ISCONFIG**.
- v** This option specifies a verbose mode where **isexplain** writes to standard output all the values for the row that will be inserted into the **ispy_explain** table in the activity database.
- V** This option displays the current version and serial number to standard output. For example:

```
INFORMIX I-Spy Version 1.00.UB11 Software Serial  
Number RDS#N000000
```

- explainfile ...** A list of explain files to process. For more information on the **sqexplain.out** file, see the SET EXPLAIN statement in the *Informix Guide to SQL: Syntax*.

The **isexplain** utility processes each explain file listed on the command line and looks for SQL statements that have the tag '--ISPY' at the beginning of a new line immediately following the text for the statement. The **ispy** daemon appends this tag to queries if the EXPLAINTAGS parameter is set to 1 in the I-Spy configuration file.

When it finds a tag, **isexplain** executes the following steps:

1. The **act_explain_id** column of the **ispy_activity** table is searched to see if an **explain_id** matching the tag exists.
2. If a matching **explain_id** value is found, a new row is inserted into the **ispy_explain** table. The **exp_text** column contains the explain text from the file.
3. Finally, the matching **act_explain_id** column of the **ispy_activity** table is updated so that the first character of that column now contains a Y instead of an N.

The first character of the **act_explain_id** column is either a Y or an N, indicating that a corresponding row exists or does not exist in the **ispy_explain** table. The remaining characters in the column are three numbers separated by a hyphen that contain a unique session number and SQL statement number, respectively. Both are generated internally by I-Spy. For example, the following values are all possible for the **act_explain_id** column:

```
N1-35-69
Y4-2-500
N0-1-1
```

isload: Load Files into the Activity Database

Use the **isload** utility to load activity that was captured to the file system into the activity database.

Syntax

```
isload -c configfile -V
```

Options

-c configfile This option specifies the full path to the configuration file that you want to use. If a configuration file is not specified, **isload** looks at the environment variable **ISPY_CONFIGFILE** for the full path. If this environment variable is not set, **isload** uses the default directory **\$ISPY_DIR/etc/isconfig**.

-V This option displays the current version and serial number to standard output. For example:

```
INFORMIX I-Spy Version 1.00.UB11 Software Serial
Number RDS#N000000
```



The **isload** utility looks for session and activity files that exist in the activity directory that the `ACTIVITY_DIR` parameter specifies.

Each I-Spy client session will create two files in the activity directory named **session.#**, and **activity.#**. The # is a unique number that I-Spy generates and is used to associate the two files to a single client session.

Warning: files to **session.#.old** and **activity.#.old** to represent that they have already been loaded. Over time you might end up with a considerable number of files with a **.old** extension in the activity directory. These files can safely be removed. The **isload** utility does not automatically remove the files after processing to ensure that any possible errors that occur can be corrected and the files can be processed again by manually removing the **.old** extension and executing **isload** again.

ispy: I-Spy Daemon Process

The **ispy** utility is the daemon process for I-Spy that listens for new client connections and routes all message traffic to the shadow database server.

Syntax

```
ispy -c configfile -d -k -m -r -v
```

Options

- c configfile** This option specifies the full path to the configuration file that you want I-Spy to use. If a configuration file is not specified, **ispy** looks at the environment variable **ISPY_CONFIGFILE** for the full path. If this environment variable is not set, **ispy** uses the default directory **\$ISPY_DIR/etc/isconfig**.
- d** This option directs **ispy** to execute without becoming a daemon process. In this mode, it does not fork and disconnect itself from the controlling terminal but continues to operate in the foreground. This option is intended to be used by Informix Technical Support, and you will likely never use it.
- Return value: None
- k** This option kills the currently running **ispy** daemon. The message:
- ```
I-Spy process 456 kill with a SIGTERM signal by ispy -k
```
- is written to standard output and also placed in the I-Spy log file. If an **ispy** daemon process is not currently running, the message
- ```
An ispy process is not currently running
```
- will be written to standard error.
- Return value: 0 if an ispy process was successfully killed, otherwise 1.
- m** This option displays the last 20 lines of the I-Spy log file to standard output.
- Return value: Always 0.
- r** This option determines if an **ispy** daemon process is currently running. If a daemon process is currently running, the process id number is written to standard output.

Return value: 0 if a daemon process is running, otherwise 1.

-V

This option displays the current version and serial number to standard output. For example:

```
INFORMIX I-Spy Version 1.00.UB11 Software Serial  
Number RDS#N000000
```

Return value: Always 0

If **ispy** is executed without any options, it becomes a daemon process and listens for new client connections. The TCP port number used for listening is obtained from the **sqlhosts** file entry for the server name that matches the ISPYSERVER value in the configuration file.

If **ispy** detects any problems in the configuration file, it will issue an appropriate error message to standard error and exit without becoming a daemon process. In this case, the return value is set to 1.



Important: For changes to the configuration file to take effect, the **ispy** daemon must be stopped and restarted. Existing client connections are not affected when **ispy** is stopped, but new client connections fail until the daemon is restarted. Also, existing client connections continue to use the previous values in the configuration file and are unaware of any changes.

Activity Database Layout

Creating the Activity Database	6-3
Determine Database Location	6-4
Determine act_sqlstmt Column Type	6-4
CHARACTER Data Type	6-4
TEXT Data Type	6-5
Prepare the ispy.sql File	6-5
Add Storage-Specific Syntax	6-5
Run ispy.sql	6-6
Grant Connect Privilege	6-6
ISPY_SESSION	6-8
ISPY_ACTIVITY	6-9
ISPY_EXPLAIN	6-13
ISPY_SQLTYPE	6-15

In This Chapter

This chapter discusses the tables involved in the activity database. This database is used to save all the SQL statements submitted by clients communicating with an Informix database server.

This chapter describes the following items:

- Creating the activity database
- The **ispy_session** table
- The **ispy_activity** table
- The **ispy_explain** table
- The **ispy_sqltype** table

Creating the Activity Database

I-Spy uses the activity database to record all the SQL statements submitted by client applications.

To create the activity database

1. Determine the location of the Informix database server where you want the activity database located.
2. Determine the column type used to store the text of SQL statements, either **CHARACTER** or **TEXT**.
3. Add storage-specific syntax for the **CREATE TABLE** statements.
4. Run **ispy.sql** to create the activity tables.
5. Grant **Connect** privilege to the activity database for appropriate users.

Determine Database Location

The activity database can be located on the same computer where I-Spy is installed, or it can be located on an entirely separate computer. It is recommended that you locate the activity database in a separate database server instance than the instance you are using I-Spy to shadow. If possible, this instance should be located on another computer. This should help to reduce the performance impact on the shadow database instance by providing dedicated resources for the activity database.

The database instance in which you create the activity database should match the server name used for the `ACTIVITY_INFORMIXSERVER` setting in your I-Spy configuration file.

Determine act_sqlstmt Column Type

The **ispy_activity** table contains a column named **act_sqlstmt** that is used to store the text of SQL statements captured by I-Spy. In the **\$ISPY_DIR/sql/ispy.sql.std** file, this column is commented out because you have to determine what column type to use, either `CHARACTER` or `TEXT`.

The length of SQL statements submitted by clients is entirely arbitrary, so it is important that you choose a column type and length that is sufficient for your purposes but also minimizes the amount of disk space used.

CHARACTER Data Type

If you choose the `CHARACTER` data type for **act_sqlstmt**, you will likely waste disk space with a very large size, or lose parts of SQL statements with a very small size. For example, a size of `CHAR(256)` is probably too small. If the length of a stored SQL statement is longer than 256 characters, it will be truncated when it is saved to the activity database and you will not be able to view the entire statement later.

If, on the other hand, you choose a size such as `CHAR(16384)`, you will waste significant amounts of disk space because each SQL statement will consume 16 kilobytes.

Informix recommends that you start with a small average size such as 512, 1024, or 2048 bytes to begin with. If you notice that SQL statements are being truncated, you can always use the SQL ALTER TABLE statement to increase the size of the column. Changing the column size later has no effect on I-Spy.

TEXT Data Type

The TEXT data type is an alternative to the CHARACTER data type for storing SQL statements. The advantage of using the TEXT type is that it is entirely variable in length just like the SQL statements your clients submit. The smallest amount of disk space is used by storing the SQL statements in a TEXT column in a regular dbspace, not a blobspace.

A blobspace always stores TEXT data on separate blobpages, which will typically result in a large amount of unused space. TEXT data types can share dbspace blobpages if more than one row can fit on a single page, or if more than one trailing portion of a TEXT column can fit on a single page. For general information on how TEXT data is stored, refer to your *Administrator's Guide*.

Prepare the ispy.sql File

To prepare the **ispy.sql** file using a standard text editor, follow these steps:

1. Copy the **ISPY_DIR/sql/ispy.sql.std** file to a new file named **ispy.sql** in the same directory.

The new file, **ispy.sql**, will be used to create all the tables for the activity database. Do not modify the **ispy.sql.std** file because it reflects the default configuration of the activity database.

2. Edit the new **ispy.sql** file and change the column type of the **act_sqlstmt** column to either TEXT or CHAR depending on the considerations previously presented.

Add Storage-Specific Syntax

The CREATE TABLE statements in **ispy.sql.std** have no syntax for storage options, lock modes, or extent sizes. If appropriate, you can add this syntax. For more information on these options, see the CREATE TABLE statement in the *Informix Guide to SQL: Syntax*.

Run ispy.sql

Use DB-Access as user **informix** to create the activity database in the database server where you want it located. You can create the database with or without logging, although it is recommended to create it without logging to reduce the performance overhead of writing SQL activity to a logged database. For example, in DB-Access you would execute the following SQL statement:

```
create database ispy ;
```

It is important that the database name you choose matches the `ACTIVITY_DBNAME` parameter setting in the I-Spy configuration file you are using. Informix recommends that you use **ispy** for the database name unless you have a specific reason to choose another name.

Warning: *Do not make the activity database ANSI compliant because I-Spy will fail to execute.*

After the database is created, change to the **\$ISPY_DIR/sql** directory and execute the **ispy.sql** script to create the activity database tables, as the following example shows:

```
$ cd $ISPY_DIR/sql
$ dbaccess ispy ispy.sql
```

Tip: *If the activity database resides on a different computer than the I-Spy installation, you might have to copy the **ispy.sql** file to this computer and run DB-Access there.*

Grant Connect Privilege

It is necessary to grant Connect privilege on the I-Spy activity database for those users who need access. In the simplest configuration, only user **informix** needs Connect privilege, which it has automatically, since you created the activity database as user **informix**. In this case, there is nothing further to do.



If you plan on using the Web interface to administer I-Spy, you must grant Connect privilege to the user that your Web server runs as. You might need to contact your Web administrator for this user name. For example, the Apache Web server uses a configuration file named **httpd.conf** that specifies this user. The following lines show the section of this file that specifies that user:

```
# User/Group: The name (or #number) of the user/group to run
# httpd as:
# On SCO (ODT 3) use User nouser and Group nogroup
# On HPUX you may not be able to use shared memory as nobody,
# and the suggested workaround is to create a user www and use
# that user.
User www
Group www
```

In this example, the user is named **www**, so you would grant Connect privilege by executing the following SQL statement in DB-Access:

```
grant connect to www ;
```

The Web interface comprises a number of CGI, or Common Gateway Interface, scripts that perform the various activities on the Web interface main menu. Your Web server executes these scripts as the user defined in the Web-server configuration settings. Since some of these scripts access the activity database, it is necessary to grant Connect privilege.

If you have other users besides **informix** or the Web interface that need access to the activity database, grant them Connect privilege as well. These users might be accessing the activity database for reporting or other informational purposes. For example, to grant Connect privilege for users **george** and **tom**, execute the following SQL statements in DB-Access.

```
grant connect to george ;
grant connect to tom ;
```

ISPY_SESSION

The **ispy_session** table describes each client connection to the shadow database server. Each row in the table indicates a single client session. The **ispy_session** table contains the columns that the following table shows.

Column Name	Type	Explanation
ses_id	SERIAL NOT NULL	Unique id for each session
ses_uname	VARCHAR(255)	User name of client
ses_hostname	VARCHAR(255)	Host name of where the client application connected from
ses_hostip	VARCHAR(255)	The ip address of the clients host. Example: '192.147.100.202'
ses_dbname	VARCHAR(255)	The database name specified by the client when it connects to the database server, otherwise null
ses_start	DATETIME YEAR TO SECOND	The date and time when the session was started
ses_end	DATETIME YEAR TO SECOND	The date and time when the session was ended
ses_abortrule	INTEGER	The rule number from the I-Spy configuration file for the rule that terminated this session, otherwise 0

ISPY_ACTIVITY

The **ispy_activity** table describes each SQL statement submitted by client applications. Each row in the table represents the execution of exactly one SQL statement for a particular server session. The **ispy_activity** table contains the columns that the following table shows.

Column Name	Type	Explanation
ses_id	INTEGER NOT NULL	A link to the ses_id column in the ispy_session table
act_id	SERIAL NOT NULL	A unique id for this SQL statement
act_sqltype	SMALLINT	A value that indicates what type of SQL statement this is. This value corresponds directly to the value of sqlca.sqlcode after a DESCRIBE statement in an INFORMIX-ESQL/C program. Labels for all possible SQL types are contained in the ispy_sqltype table.
act_sqlstmt	Either CHAR() or TEXT	The text of this SQL statement
act_isdistrib	SMALLINT	A flag indicating if this SQL statement is distributed, that is, from another database server and not a client. 0 = no, 1 = Yes. This functionality is currently supported. The value should always be 0.
act_curname	VARCHAR(255)	The name of the cursor used for this SQL statement. Will be null if no cursor name was defined.
act_curishold	SMALLINT	A flag indicating if the cursor was declared with hold. 0 = No, 1 = Yes.
act_curisscroll	SMALLINT	A flag indicating if the cursor was declared as scrolling. 0 = No, 1 = Yes.
act_prepare	DATETIME YEAR TO SECOND	The date and time when this statement was prepared

(1 of 3)

Column Name	Type	Explanation
act_starttime	DATETIME YEAR TO SECOND	The date and time when this statement began executing
act_curopen	DATETIME YEAR TO SECOND	The date and time when the cursor for this statement was opened
act_firstrow	DATETIME YEAR TO SECOND	The date and time when the first row of the cursor for this statement was fetched from the database server
act_lastrow	DATETIME YEAR TO SECOND	The date and time when the last row of the cursor for this statement was fetched from the database server
act_curclose	DATETIME YEAR TO SECOND	The date and time when the cursor for this statement was closed
act_endtime	DATETIME YEAR TO SECOND	The date and time when this statement finished executing
act_enginetime	INTERVAL HOUR(4) TO FRACTION(2)	The amount of time the database server spent processing this statement
act_estrows	INTEGER	The estimated number of rows this statement will return
act_cost	INTEGER	The estimated cost to execute this statement
act_nrows	INTEGER	The total number of rows returned by this statement. For an insert cursor, this indicates the total number of rows sent to the database server
act_rowlen	SMALLINT	The length of the result set in bytes for this statement
act_blobbytes	FLOAT	The total number of TEXT bytes returned in all rows of this statement
act_nfetch	INTEGER	The number of fetch requests the client sent to the database server

(2 of 3)

Column Name	Type	Explanation
act_sqlerr	INTEGER	The error returned by the database server for this statement. This value will be 0 if there was no error.
act_isamerr	INTEGER	The isam error code returned by the database server for this statement. This value will be 0 if there was no error.
act_abortrule	INTEGER	The rule number from the I-Spy configuration file that caused this statement to be stopped, or 0 if no rule was involved.
act_explain_id	VARCHAR(25)	A link into the ispy_explain table where the explain information for this statement might be located

(3 of 3)

It is important to understand how clients execute SQL statements and how I-Spy reflects this in populating the **ispy_activity** table. Simple SQL statements in client applications that are not prepared, and have no cursor associated with them, result in a single row each time they are executed.

SQL statements that are prepared and executed result in a single row for each execution of the prepared statement, with the text of the SQL statement contained in each row. For example, consider the following lines in an INFORMIX-ESQL/C program.

```
exec sql prepare p1 from "delete from customer where cust_num
= 100" ;
exec sql execute p1 ;
exec sql execute p1 ;
exec sql execute p1 ;
```

This example will result in three rows in the **ispy_activity** table, where each **act_sqlstmt** column contains the text "delete from customer where cust_num = 100". Each row will have a unique **act_id** value, and they are not related in any way in the table.

SQL statements with cursors operate in the same manner. Each time a cursor is opened and closed, a new row is written to the **ispy_activity** table.

If the **act_prepare** column is null, it indicates that the client never prepared the SQL statement. If no cursor was associated with the SQL statement, the **act_curname**, **act_curopen**, and **act_curclose** columns are all null.

The elapsed time that the SQL statement took to execute is **act_endtime - act_starttime**. This includes any time the client spent processing with a cursor open but not waiting for a response from the database server. For example, a client application could open a cursor for an SQL statement that returns 100, fetch the first 10 rows from the database server, and then perform some other type of processing. After this extra processing concludes, the application can fetch the remaining 90 rows. This extra processing time is reflected in the **act_endtime** column because it includes all the time while the cursor was open.

The **act_enginetime** column is an accumulation of the engine time spent processing a client request. In the preceding example, the extra processing time spent by the application would not be included in the **act_enginetime** column. For most applications, the **act_enginetime** should be very close to the **act_endtime - act_starttime**. A significant difference between these values indicates that the application is spending the time performing nondatabase work. You might want to explore this area to explore if application performance is a problem.

The **act_explain_id** column is a link into the **ispy_explain** table, which might contain the explain text for the query. The first character of this column is either a Y or an N indicating that a corresponding row exists or does not exist in the **ispy_explain** table. When the **isexplain** utility loads the explain text for a query, it updates the first character of this column changing the N to a Y. The remaining characters in the column are three numbers separated by a hyphen, which contain an activity id, a unique session number, and an SQL statement number, respectively. I-Spy generates these three numbers internally. For example, the following values are all possible for the **act_explain_id** column.

```
N0-35-69  
Y3-2-500  
N10-1-1
```

ISPY_EXPLAIN

The **ispy_explain** table describes the SQL explain text for queries contained in the **ispy_activity** table. Each row of this table corresponds to exactly one row in the **ispy_explain** table. The **ispy_explain** table contains the columns that the following table shows.

Column Name	Type	Explanation
act_explain_id	VARCHAR(25)	A link to the act_explain_id column of the act_activity table
exp_text	TEXT	The explain text for this SQL statement
exp_seqscan	SMALLINT	The number of times the string “SEQUENTIAL SCAN” appears in the explain text
exp_autoindex	SMALLINT	The number of times the string “AUTOINDEX PATH” appears in the explain text
exp_indexpath	SMALLINT	The number of times the string “INDEX PATH” appears in the explain text
exp_mindexpath	SMALLINT	The number of times the string “MULTI INDEX PATH” appears in the explain text
exp_keyonly	SMALLINT	The number of times the string “(Key-Only)” appears in the explain text
exp_aggregate	SMALLINT	The number of times the string “(Aggregate)” appears in the explain text
exp temptable	SMALLINT	The number of times the string “Temporary File Required For:”, or “(Temp Table for View)” appears in the explain text

(1 of 2)

Column Name	Type	Explanation
exp_sortscan	SMALLINT	The number of times the string “SORT SCAN:” appears in the explain text.
exp_mergejoin	SMALLINT	The number of times the string “MERGE JOIN” appears in the explain text
exp_hashjoin	SMALLINT	The number of times the string “DYNAMIC HASH JOIN” appears in the explain text
exp_buildinner	SMALLINT	The number of times the string “(Build Inner)” appears in the explain text
exp_buildinnerb	SMALLINT	The number of times the string “(Build Inner Broadcast)” appears in the explain text
exp_buildouter	SMALLINT	The number of times the string “(Build Outer)” appears in the explain text
exp_buildouterb	SMALLINT	The number of times the string “(Build Outer Broadcast)” appears in the explain text
exp_unionquery	SMALLINT	The number of times the string “Union Query” appears in the explain text
exp_subquery	SMALLINT	The number of times the string “Subquery:” appears in the explain text.

(2 of 2)

The **isexplain** utility adds new rows to this table by processing **sqexplain.out** files created by an Informix database server. This is the only manner in which this table is populated. The **ispy** daemon process never accesses this table.

The columns in this table, such as **exp_seqscan**, count the number of specific strings in the explain text. You can use these columns to quickly locate the SQL statements that might be performance problems or that contain a behavior that you want to investigate. For example, you might be interested in the queries that contain the string “AUTOINDEX PATH” because this string is probably an indication that an index is needed. You could use the following SELECT statement to find those queries:

```
select
    exp_text
from
    ispy_explain
where
    exp_autoindex > 0
```

If you wanted further information on these queries from the activity table, you could execute the following statement:

```
select
    e.exp_text,
    a.*
from
    ispy_activity a,
    ispy_explain e
where
    e.act_explain_id = a.act_explain_id and
    e.exp_autoindex > 0
```

ISPY_SQLTYPE

The **ispy_sqltype** table describes the various types of SQL statements that you can submit to a database server. Each row contains a unique identifier for a specific type of SQL statement and a short name that describes it. The **ispy_sqltype** table contains the columns that the following table shows.

Column Name	Type	Explanation
sql_id	SMALLINT	Link to the act_sqltype column in the ispy_activity table
sql_name	VARCHAR(255)	Short descriptive name for this type of SQL statement

The Web interface uses this reference table to provide descriptive names for SQL statement types. The table is populated by the file **\$ISPY_DIR/sql/ispysqltype.unl** when the **\$ISPY_DIR/sql/ispysql** file is processed.

Currently only the reports in the **\$ISPY_DIR/sql** directory and the Web interface access this table.

The following table displays the data contained in the **ispysqltype** table.

sql_id	sql_name
1	DATABASE
2	SELECT
3	SELINTO
4	UPDATE
5	DELETE
6	INSERT
7	UPDCURR
8	DELCURR
9	LDINSERT
10	LOCK
11	UNLOCK
12	CREADB
13	DROPDB
14	CRETAB
15	DRPTAB
16	CREIDX
17	DRPIDX
18	GRANT

(1 of 4)

sql_id	sql_name
19	REVOKE
20	RENTAB
21	RENCOL
22	CREAUD
23	STRAUD
24	STPAUD
25	DRPAUD
26	RECTAB
27	CHKTAB
28	REPTAB
29	ALTER
30	STATS
31	CLSDB
32	DELALL
33	UPDALL
34	BEGWORK
35	COMMIT
36	ROLLBACK
37	SAVEPOINT
38	STARTDB
39	RFORWARD
40	CREVIEW
41	DROPVIEW
42	DEBUG

(2 of 4)

sql_id	sql_name
43	CREASYN
44	DROPSYN
45	CTEMP
46	WAITFOR
47	ALTIDX
48	ISOLATE
49	SETLOG
50	EXPLAIN
51	SCHEMA
52	OPTIM
53	CREPROC
54	DRPPROC
55	CONSTRMODE
56	EXECPROC
57	DBGFILE
58	CREOPCL
59	ALTOPCL
60	DRPOPCL
61	OPRESERVE
62	OPRELEASE
63	OPTIMEOUT
64	PROCSTATS
67	SKINHIBIT
68	SKSHOW

(3 of 4)

sql_id	sql_name
69	SKSMALL
70	CRETRIG
71	DRPTRIG
72	UNKNOWN
73	SETDATASKIP
74	PDQPRIORITY
75	ALTFRAG
76	SETOBJMODE
77	START
78	STOP
79	SETMAC
80	SETDAC
81	SETTBLHI
82	SETLVEXT
83	CREATEROLE
84	DROPROLE
85	SETROLE
86	PASSWD
87	RENDB

(4 of 4)

Using the Web Interface

The Main Window	7-4
The Startup Menu Option	7-5
The Shutdown Menu Option	7-6
The Config File Menu Option	7-7
The Clients Menu Option	7-8
The Reports Menu Option	7-10
The View Log Menu Option.	7-12
The User Manual Menu Option	7-12

In This Chapter

This chapter describes how to use the various options of the I-Spy Web interface. You should have already configured the Web interface following the directions in [Chapter 4, “Configuring the Web Interface.”](#)

This chapter covers the following topics:

- The Main Screen
- The **Startup** Menu Option
- The **Shutdown** Menu Option
- The **Config File** Menu Option
- The **Clients** Menu Option
- The **Reports** Menu Option
- The **View Log** Menu Option
- The **User Manual** Menu Option

The Main Window

To display the main window of the Web interface, open the URL `http://hostname/ispay/` in your Web browser, where *hostname* is the name of the computer where you have installed I-Spy. [Figure 7-1](#) shows the main window.

Figure 7-1
Main Window

The interface consists of three frames, one at the top with the product name, one with all the menu options along the left side, and a third frame that changes depending on the menu option selected.

Status information on I-Spy appears when you click the **Status** button from the menu or when the opening page is first loaded in your Web browser. The status window contains the following fields.

Field	Description
Host	The host name of the computer where I-Spy is installed
Version	The version and serial number of I-Spy. This is the same information that the command 'ispy -V' generates
Configuration File	The full path to the configuration file that I-Spy uses
Status	The ispy daemon process currently running

The Startup Menu Option

The **Startup** menu option takes you to the window in [Figure 7-2](#), which allows you to start the **ispy** daemon.

Figure 7-2
Startup Window

Click the **Startup** button to run the **ispy** daemon process. After the daemon is started, the Status window appears.

The Shutdown Menu Option

The **Shutdown** menu option takes you to the window in [Figure 7-3](#), which allows you to stop the **ispy** daemon from running.

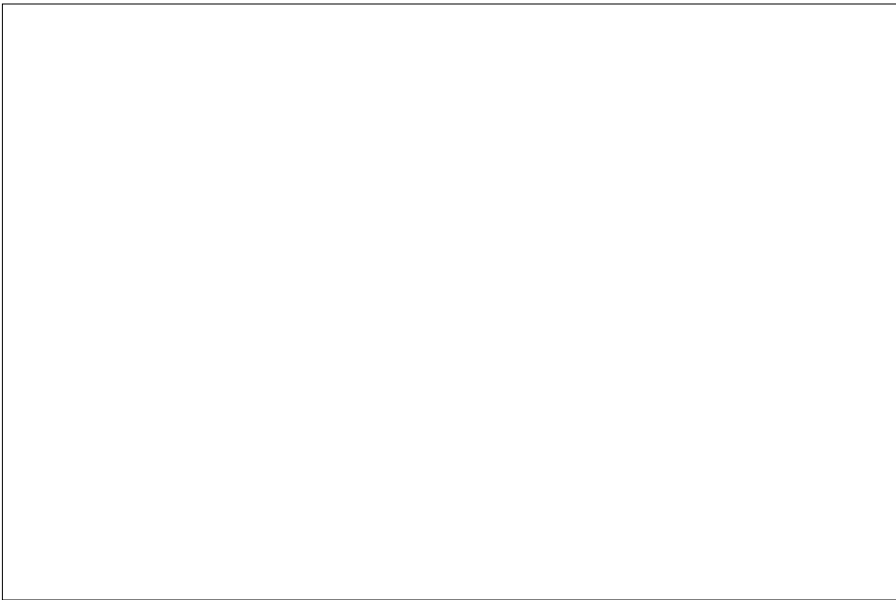


Figure 7-3
Shutdown Window

Click the **Shutdown** button to execute the command **ispy -k** to kill the **ispy** daemon process. After the daemon process shuts down, the Status window appears.

If you shut down I-Spy, client applications will no longer be able to connect to the ISPYSERVER value specified in the configuration file. However, clients that are currently connected will continue to operate without problems.

The Config File Menu Option

The **Config File** menu option takes you to the window in [Figure 7-4](#), which allows you to edit the values in the I-Spy configuration file.

Figure 7-4
*Edit Configuration
File Window*

Click the **Submit Changes** button to write the contents of the text window to the configuration file and save any changes you have made. The current configuration file is copied to a file of the same name but with a **.old** extension before the new changes are written.

The Clients Menu Option

The **Clients** menu option takes you to the window in [Figure 7-5](#), which displays a list of all clients currently connected to I-Spy.

Figure 7-5
Clients Window

Each client name is a hot link that you can select to provide more detailed information on that particular client. [Figure 7-6](#) shows a Client Detail window that is presented when you select a hot link.

Figure 7-6
*Client Detail
Window*

The Client Detail window displays the last five SQL statements that a client has executed. If the client has a currently executing SQL statement, it is not displayed.

The Reports Menu Option

The **Reports** menu option takes you to the window in [Figure 7-7](#), which displays a list of reports that you can execute against the activity database.

Figure 7-7
Reports Window

I-Spy prompts for a number of input values before it generates the report. [Figure 7-8](#) shows the values required to execute the *Longest running queries* report.

Figure 7-8
*Longest Running
Queries Report*

Each report has a similar set of fields that you must fill in before you click the **Run Query** button on the bottom of the screen to execute the report.

Each report has the following set of common input fields that control how the report is returned.

Field	Description
How many rows to return?	The maximum number of rows to return for this report
Output format	The output format for the report. HTML returns the report as a table, while unload returns the report with one row per line and a specific delimiter between each field. The unload format is intended to be used to import the results into other applications, such as a spreadsheet.

Field	Description
Unload field delimiter	A single character used to separate fields if the output format chosen is unload.
Show the SQL statement used for this report?	If this box is checked, the SQL statement executed against the activity database is included with the report results. This is useful if you want to create your own reports and see how they were generated.

The View Log Menu Option

The **View Log** menu option displays the last 20 lines of the I-Spy log file in a new window, as [Figure 7-9](#) shows.

Figure 7-9
View Log Window

This window displays the output of the **ispy -m** command.

The User Manual Menu Option

The **User Manual** menu option opens the **.pdf** file that contains the *Informix I-Spy User Manual*. The Adobe Acrobat Reader must be installed to view this file.

Common Questions

Common Questions About I-Spy	8-3
How do I prevent users from returning more than 1000 rows in a single query?	8-3
How do I send an email to the database administrator when a query has returned more than 1000 rows without ending the query? 8-4	
How do I prevent all users except informix from returning more than 1000 rows?	8-6
How do I assign a default PDQ priority for all users?	8-6
How do I check the syntax of the rules in my configuration file without shutting down and starting up I-Spy?	8-7

In This Chapter

This chapter provides additional information on a number of practical issues.

Common Questions About I-Spy

The following sections answer these questions:

- How do I prevent users from returning more than 1000 rows in a single query?
- How do I send an email to the database administrator when a query has returned more than 1000 rows without ending the query?
- How do I prevent all users except **informix** from returning more than 1000 rows?
- How do I assign a default PDQ priority for all users?
- How do I check the syntax of the rules in my configuration file without shutting down and starting up I-Spy?

How do I prevent users from returning more than 1000 rows in a single query?

Create a rule with a rows condition in the active phase of the configuration file. For example:

```
PHASE ACTIVE

RULE 10
ROWS 1000
ACTION ERROR Row limit 1000 exceeded
```

The database server returns rows to the client in groups, rather than one at a time. This results in queries not being stopped at exactly 1000 rows, but within one grouping of rows. For example, if the database server is returning 23 rows at a time, the 1st group will contain rows 1-23, the 2nd rows 24-46, on to the 43rd group containing rows 967-989, and the 44th containing 990-1012. With the above rule, I-Spy will prevent further rows from returning, but the client will see all rows up to and including 1012.

How do I send an email to the database administrator when a query has returned more than 1000 rows without ending the query?

Use the EXECUTE action to run a shell script that sends an appropriate error message. For example, the following rule could be used.

```
PHASE ACTIVE
RULE 10
ROWS 1000
ACTION EXECUTE /usr/ispys/bin/ispymail.sh
```

The script **ispymail.sh** is included in the I-Spy distribution and has the following text.

```
#!/bin/sh

MAILTO='dba@company.com'
SUBJECT='I-Spy Alert'

echo '

This email is being generated from the script
$ISPY_EXECUTE.

USERNAME          $ISPY_USERNAME
CLIENTHOST        $ISPY_CLIENTHOST
CLIENTIP          $ISPY_CLIENTIP
DBNAME            $ISPY_DBNAME
RULE              $ISPY_RULE
SQL               $ISPY_SQL
ECOST             $ISPY_ECOST
EROWS             $ISPY_EROWS
ROWS             $ISPY_ROWS
NFETCH           $ISPY_NFETCH
START            $ISPY_START
PREPARE          $ISPY_PREPARE

" | mailx -s '$SUBJECT' $MAILTO
```

You can use this script as a template to create a more specific one for your environment.

How do I prevent all users except informix from returning more than 1000 rows?

Create two rules in the active phase of the configuration file. The first rule includes a user condition for **informix**, while the second rule does not.

```
PHASE ACTIVE

RULE 1
USER informix
ROWS 99999999
ACTION ERROR Row limit exceeded

RULE 2
ROWS 1000
ACTION ERROR Row limit of 1000 exceeded
```

Be sure to include the rules in the configuration file in the order shown because I-Spy processes rules in this order. User **informix** will match rule number 1 and allow an arbitrarily large number of rows to be returned, while all other users will mat

How do I assign a default PDQ priority for all users?

This question has two answers, and both have different ramifications. First, you could create a rule in the database-open phase of the configuration file. For example,

```
PHASE DBOPEN
RULE 1
ACTION EXECSQL set pdqpriority 30
```

This rule sets the **pdqpriority** when a database is first opened, but the user can change it later. Alternatively, you could create the following rule in the active phase of the configuration file:

```
PHASE ACTIVE
RULE 2
ACTION EXECSQL set pdqpriority 30
```

This rule sets the **pdqpriority** before every SQL statement that executes. This prevents users from changing the **pdqpriority** themselves.

How do I check the syntax of the rules in my configuration file without shutting down and starting up I-Spy?

The command **ispy -r**, which determines if an **ispy** process is running, also checks the syntax of the configuration file and reports any errors. This report provides a simple way to verify the contents of the configuration file before shutting down and restarting the **ispy** process.

For example, if you had the following syntax error in your configuration file:

```
PHASE active10
```

ispy -r would report:

```
Line 18: Invalid PHASE argument <active10>
```


Index

A

access.conf 4-6, 4-9
ACTION 3-8
Activity database
 connect privilege 6-5
 creating 6-3
 ispy.sql schema file 6-5
 ispy_activity table 6-9
 ispy_explain table 6-13
 ispy_session table 6-8
 ispy_sqltype table 6-15
ACTIVITY_CAPTURE 3-14
ACTIVITY_DBNAME 3-15
ACTIVITY_DIR 3-15
ACTIVITY_ID 3-16
ACTIVITY_INFORMIXSERVER
 3-16
ANSI compliance
 level Intro-12
Apache web server 4-3

B

Boldface type Intro-6

C

Clients
 configuring INFORMIX-
 Connect 2-13
 general configuration 2-13
 no changes 3-30
Code, sample, conventions
 for Intro-10

Command-line conventions
 elements of Intro-8
 example diagram Intro-9
 how to read Intro-9
Comment icons Intro-7
Compliance with industry
 standards Intro-12
Configuration file
 general syntax 3-3
 rule execution order 3-6
 rules overview 3-4
Contact information Intro-13
Conventions,
 documentation Intro-5
custom_reports.html 4-10

D

DEBUGFILE 3-17
DEBUGLEVEL 3-17
Default locale Intro-5
Dependencies, software Intro-4
Documentation, types of
 documentation notes Intro-12
 error message files Intro-11
 on-line manuals Intro-11
 printed manuals Intro-11
 release notes Intro-12

E

ECOST 3-18
ENDSESS 3-9

Environment variables Intro-6
 INFORMIXDIR 2-8
 INFORMIXSQLHOSTS 2-8
 ISPY_CONFIGFILE 2-7
 ISPY_DIR 2-7
 overview of 2-6
 PATH 2-9
 shared-library search path 2-8
 en_us.8859-1 locale Intro-5
 EROWS 3-18
 ERROR 3-9
 Error message files Intro-11
 EXECSQL 3-10
 EXECUTE 3-10
 EXPLAINTAGS 3-12, 3-19

F

Find Error utility Intro-11
 finderr utility Intro-11

G

Global Language Support
 (GLS) Intro-5

H

HOST 3-20
 httpd.conf 6-7

I

Icons
 Important Intro-7
 Tip Intro-7
 Warning Intro-7
 Important paragraphs, icon
 for Intro-7
 index.html 4-6
 Industry standards, compliance
 with Intro-12
 isexplain 5-3
 isload 5-5
 ISO 8859-1 code set Intro-5
 ispy 5-6
 ispymail.sh 3-12, 8-5

ISPYSERVER 3-21
 ispy_activity table 6-9
 ispy_explain table 6-13
 ispy_session table 6-8
 ispy_sqltype table 6-15
 ITIME 3-22

L

Limitations, product 1-5
 Locale Intro-5
 LOGFILE 3-22

M

Message file for error
 messages Intro-11

N

Netscape Web server 4-12
 NOLOG 3-13

O

On-line manuals Intro-11

P

PDQ priority 8-6
 PHASE 3-23
 Printed manuals Intro-11

Q

QTIME 3-23

R

Release notes Intro-12
 rofferr utility Intro-11
 ROWS 3-25
 RULE 3-25

S

Sample-code conventions Intro-10
 Setnet32 2-13
 SHADOWSERVER 3-27
 Software dependencies Intro-4
 SQL 3-27
 SQL code Intro-10
 sqlhosts
 ACTIVITY_INFORMIXSERVER
 3-17
 ISPYSERVER 3-21
 locating the file 2-8
 preparing the file 2-11
 SHADOWSERVER 3-27
 SQLENGTH 3-28
 SQLTRIM 3-28
 srm.conf 4-5, 4-6
 System requirements
 database Intro-4
 software Intro-4

T

Tip icons Intro-7

U

USER 3-30
 Users, types of Intro-4
 Utilities
 isexplain 5-3
 isload 5-5
 ispy 5-6

W

Warning icons Intro-7
 Web interface
 Apache web server 4-3
 Clients menu option 7-8
 Config File menu option 7-7
 configuring security 4-8
 database Connect privilege 4-7
 main window 7-4
 Netscape web server 4-12
 preparing web.cfg 4-4

Reports menu option 7-10
Shutdown menu option 7-6
Startup menu option 7-5
Status menu option 7-4
User Manual menu option 7-12
View Log menu option 7-12

X

XA Protocol 1-5
X/Open compliance level Intro-12

